MVAPICH2 1.5 User Guide

MVAPICH Team

Network-Based Computing Laboratory
Department of Computer Science and Engineering
The Ohio State University

http://mvapich.cse.ohio-state.edu

Last revised: July 20, 2010

Contents

1 Overview of the MVAPICH Project

InfiniBand, 10GbE/iWARP and RDMA over Converged Ethernet (RoCE) are emerging as high-performance networking technologies to deliver low latency and high bandwidth. They are also achieving widespread acceptance due to their *open standards*.

MVAPICH (pronounced as "em-vah-pich") is an *open-source* MPI software to exploit the novel features and mechanisms of these networking technologies and deliver best performance and scalability to MPI applications. This software is developed in the Network-Based Computing Laboratory (NBCL), headed by Prof. Dhabaleswar K. (DK) Panda.

Currently, there are two versions of this MPI library: MVAPICH with MPI-1 semantics and MVAPICH2 with MPI-2 semantics. This *open-source* MPI software project started in 2001 and a first high-performance implementation was demonstrated at Supercomputing '02 conference. After that, this software has been steadily gaining acceptance in the HPC, InfiniBand, 10GigE/iWARP and RoCE communities. As of July 11, 2010, more than 1,185 organizations (National Labs, Universities and Industry) world-wide (in 59 countries) have registered as MVAPICH/MVAPICH2 users at MVAPICH project web site. There have also been more than 43,000 downloads of MVAPICH/MVAPICH2 software from the MVAPICH project site directly. In addition, many InfiniBand, 10GigE/iWARP and RoCE vendors, server vendors, systems integrators and Linux distributors have been incorporating MVAPICH/MVAPICH2 into their software stacks and distributing it. MVAPICH and MVAPICH2 are also available with the Open Fabrics Enterprise Distribution (OFED) stack. Both MVAPICH and MVAPICH2 distributions are available under BSD licensing.

Several InfiniBand systems using MVAPICH/MVAPICH2 have obtained positions in the TOP 500 ranking. The June '10 list includes the following systems: 6th ranked Pleiades at NASA with 81,920-cores; 7th ranked Tianhe-1 system at NUDT, China with 71,680-cores; 11th ranked Ranger at Texas Advanced Computing Center (TACC) with 62,976 cores; 34th ranked Juno at Lawrence Livermore National Laboratory (LLNL) with 18,224-cores; and 57th ranked Chinook at Pacific Northwest National Laboratory (PNNL) with 18,176-cores.

More details on MVAPICH/MVAPICH2 software, users list, mailing lists, sample performance numbers on a wide range of platforms and interconnects, a set of OSU benchmarks, related publications, and other InfiniBand- and iWARP-related projects (parallel file systems, storage, data centers) can be obtained from our website:`http://mvapich.cse.ohio-state.edu`

This document contains necessary information for MVAPICH2 users to download, install, test, use, tune and troubleshoot MVAPICH2 1.5 . We continuously fix bugs and update update this document as per user feedback. Therefore, we strongly encourage you to refer to our web page for updates.

2 How to use this User Guide?

This guide is designed to take the user through all the steps involved in configuring, installing, running and tuning MPI applications over InfiniBand using MVAPICH2 1.5 .

In Section 3 we describe all the features in MVAPICH2 1.5 . As you read through this section, please note our new features (highlighted as NEW) compared to the 1.4.1 version. Some of these features are designed in order to optimize specific type of MPI applications and achieve greater scalability. Section 4 describes in detail the configuration and installation steps. This section enables the user to identify specific compilation flags which can be used to turn some of the features on of off. Basic usage of MVAPICH2 is explained in Section 5. Section 6 provides instructions for running MVAPICH2 with some of the advanced features. Section 7 provides how to obtain version and related information of the library. Section 8 describes the usage of the OSU Benchmarks. If you have any problems using MVAPICH2, please check Section 10 where we list some of the common problems people face. In Section 9 we suggest some tuning techniques for multi-thousand node clusters using some of our new features. Finally in Sections 11 and 12, we list all important run-time parameters, their default values and a small description of what that parameter stands for.

3 MVAPICH2 1.5 Features

MVAPICH2 (MPI-2 over InfiniBand) is an MPI-2 implementation based on MPICH2 ADI3 layer. It also supports all MPI-1 functionalities. MVAPICH2 1.5 is available as a single integrated package (with MPICH2 1.2.1p1). This version is compliant with the latest MPI 2.2 standard.

The current release supports eight different underlying transport interfaces, as shown in Figure 1. In addition to the previous CH3-based interfaces, new interfaces based on MPICH2-Nemesis are introduced with this release.

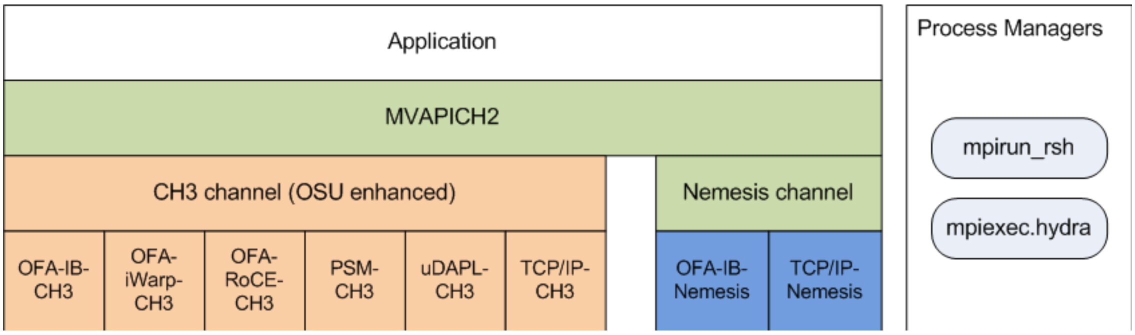

Figure 1: Overview of different available interfaces of the MVAPICH2 library.

- OFA-IB-CH3: This interface supports all InfiniBand compliant devices based on the OpenFabrics Gen2 layer. This interface has the most features and is most widely used. For example, this interface can be used over all Mellanox InfiniBand adapters, IBM eHCA adapters and QLogic adapters.
- (NEW) OFA-IB-Nemesis: This interface supports all InfiniBand compliant devices based on the OpenFabrics libibverbs layer with the emerging Nemesis channel of the MPICH2 stack. This interface can be used by all Mellanox InfiniBand adapters.
- OFA-iWARP-CH3: This interface supports all iWARP compliant devices supported by OpenFabrics. For example, this layer supports Chelsio T3 adapters with the native iWARP mode.

- OFA-RoCE-CH3: This interface supports the emerging RoCE (RDMA over Converged Ethernet) interface for Mellanox ConnectX-EN adapters with 10GigE switches.

- PSM-CH3: This interface provides native support for InfiniPath adapters from QLogic over PSM interface. It provides high-performance point-to-point communication for both one-sided and two-sided operations.

- uDAPL-CH3: This interface supports all network-adapters and software stacks which implement the portable DAPL interface from the DAT Collaborative. For example, this interface can be used over all Mellanox adapters, Chelsio adapters and NetEffect adapters. It can also be used with Solaris uDAPL-IBTL implementation over InfiniBand adapters.

- TCP/IP-CH3: The standard TCP/IP interface (provided by MPICH2) to work with a range of network adapters supporting TCP/IP interface. This interface can be used with IPoIB (TCP/IP over InfiniBand network) support of InfiniBand also. However, it will not deliver good performance/scalability as compared to the other interfaces.

- (NEW) TCP/IP-Nemesis: The standard TCP/IP interface (provided by MPICH2 Nemesis channel) to work with a range of network adapters supporting TCP/IP interface. This interface can be used with IPoIB (TCP/IP over InfiniBand network) support of InfiniBand also. However, it will not deliver good performance/scalability as compared to the other interfaces.

Please note that the support for VAPI interface has been deprecated since MVAPICH2 1.2 because OpenFabrics interface is getting more popular. MVAPICH2 users still using VAPI interface are strongly requested to migrate to the OpenFabrics-IB interface.

MVAPICH2 1.5 delivers better performance (especially with one-copy intra-node communication support with LiMIC2) compared to MVAPICH 1.2, the latest release package of MVAPICH supporting MPI-1 standard. MVAPICH2 1.5 is compliant with MPI 2.2 standard. In addition, MVAPICH2 1.5 provides support and optimizations for other MPI-2 features, multi-threading and fault-tolerance (Checkpoint-restart). A complete set of features of MVAPICH2 1.5 are indicated below. New features compared to 1.4.1 are indicated as (NEW).

- (NEW) MPI-2.2 standard compliance

- (NEW) OFA-IB-Nemesis interface design
  - OpenFabrics InfiniBand network module support for MPICH2 Nemesis modular design
  - Support for high-performance intra-node shared memory communication provided by the Nemesis design
  - Adaptive RDMA Fastpath with Polling Set for high-performance inter-node communication
  - Shared Receive Queue (SRQ) support with flow control, uses significantly less memory for MPI library
  - Header caching
  - Advanced AVL tree-based Resource-aware registration cache

  - Memory Hook Support provided by integration with ptmalloc2 library. This provides safe release of memory to the Operating System and is expected to benefit the memory usage of applications that heavily use malloc and free operations.
  - Support for TotalView debugger
  - Shared Library Support for existing binary MPI application programs to run
  - ROMIO Support for MPI-IO
  - Support for additional features (such as hwloc, hierarchical collectives, one-sided, multi-threading, etc.), as included in the MPICH2 1.2.1p1 Nemesis channel

- (NEW) Flexible process manager support
  - mpirun_rsh to work with any of the eight interfaces (CH3 and Nemesis channel-based) including OFA-IB-Nemesis, TCP/IP-CH3 and TCP/IP-Nemesis
  - Hydra process manager to work with any of the eight interfaces (CH3 and Nemesis channel-based) including OFA-IB-CH3, OFA-IWARP-CH3, OFA-ROCE-CH3 and TCP/IP-CH3

- CH3-Level design for scaling to multi-thousand nodes with highest performance and reduced memory usage
  - Support for MPI-2 Dynamic Process Management on InfiniBand clusters
  - eXtended Reliable Connection (XRC) support
  - Multiple CQ-based design for Chelsio 10GigE/iWARP
  - Multi-port support for Chelsio 10GigE/iWARP
  - Enhanced iWARP design for scalability to higher process count
  - Scalable and robust daemon-less job startup
    - Enhanced and robust mpirun_rsh framework (non-MPD-based) to provide scalable job launching on multi-thousand core clusters
    - Hierarchical ssh to nodes to speedup job start-up
    - MPMD job launch capability
    - Available for OpenFabrics (IB, iWARP and RoCE) and uDAPL interfaces (including Solaris)
  - On-demand Connection Management: This feature enables InfiniBand connections to be setup dynamically, enhancing the scalability of MVAPICH2 on clusters of thousands of nodes
    - Native InfiniBand Unreliable Datagram (UD) based asynchronous connection management for OpenFabrics Gen2-IB interface
    - RDMA CM based on-demand connection management for OpenFabrics Gen2-iWARP and OpenFabrics Gen2-IB interfaces
    - uDAPL on-demand connection management based on standard uDAPL interface
  - Message coalescing support to enable reduction of per Queue-pair send queues for reduction in memory requirement on large scale clusters. This design also increases the small message messaging rate significantly. Available for OpenFabrics Gen2-IB interface

  - Hot-Spot Avoidance Mechanism (HSAM) for alleviating network congestion in large scale clusters. Available for OpenFabrics Gen2-IB interface
  - RDMA Read utilized for increased overlap of computation and communication for OpenFabrics device. Available for OpenFabrics Gen2-IB and iWARP interfaces
  - Shared Receive Queue (SRQ) with flow control. This design uses significantly less memory for MPI library. Available for OpenFabrics Gen2-IB interface.
  - Adaptive RDMA Fast Path with Polling Set for low-latency messaging. Available for OpenFabrics Gen2-IB and iWARP interfaces.
  - Enhanced scalability for RDMA-based direct one-sided communication with less communication resource. Available for OpenFabrics (IB and iWARP) interfaces.

- Dynamic Process Management (DPM). Available for OpenFabrics IB interface.

- Fault tolerance support

  - Checkpoint-restart support for application transparent systems-level fault tolerance. BLCR-based support using OpenFabrics Gen2-IB interface.
    - Scalable Checkpoint-restart with mpirun_rsh framework
    - Scalable Checkpoint-restart with Fault Tolerance Backplane (FTB) framework (FTB-CR)
    - Checkpoint-restart with intra-node shared memory (user-level) support
    - Checkpoint-restart with intra-node shared memory (kernel-level with LiMIC2) support
    - Allows best performance and scalability with fault-tolerance support
  - Application-initiated system-level checkpointing is also supported. User application can request a whole program checkpoint synchronously by calling special MVAPICH2 functions.
    - Flexible interface to work with different files system. Tested with ext3 (local disk), NFS and PVFS2
  - Network-Level fault tolerance with Automatic Path Migration (APM) for tolerating intermittent network failures over InfiniBand

- Enhancement to software installation

  - Automatically detects system architecture and adapter types and optimizes MVAPICH2 for any particular installation
  - An utility (mpiname) for querying the MVAPICH2 library version and configuration information

- Optimized intra-node communication support by taking advantage of shared-memory communication. Available for all interfaces.

  - Kernel-level single-copy intra-node communication solution based on LiMIC2

        - ∗ LiMIC2 is designed and developed by System Software Laboratory at Konkuk University, Korea
    - Efficient Buffer Organization for Memory Scalability of Intra-node Communication
    - Multi-core optimized
    - Portable Hardware Locality (hwloc) support for defining CPU affinity
    - (NEW) Efficient CPU binding policies (bunch and scatter) to specify CPU binding per job for modern multi-core platforms
    - Also allows user-defined flexible processor affinity
    - Optimized for Bus-based SMP and NUMA-Based SMP systems
    - Efficient support for diskless clusters

- Shared memory optimizations for collective communication operations. Available for all interfaces.
    - K-nomial tree-based solution together with shared memory-based broadcast for scalable MPI_Bcast operations
    - Optimized and tuned MPI_Alltoall
    - Efficient algorithms and optimizations for barrier, reduce and all-reduce operations

- Integrated multi-rail communication support. Available for OpenFabrics Gen2-IB and iWARP interfaces.
    - Multiple queue pairs per port
    - Multiple ports per adapter
    - Multiple adapters
    - Support for both one-sided and point-to-point operations
    - Support for OpenFabrics Gen2-iWARP interface and RDMA CM (for Gen2-IB).

- Multi-threading support. Available for all interfaces, including TCP/IP.

- High-performance optimized and scalable support for one-sided communication: Put, Get and Accumulate. Supported synchronization calls: Fence, Active Target, Passive (lock and unlock). Available for all interfaces.
    - Direct RDMA based One-sided communication support for OpenFabrics Gen2-iWARP and RDMA CM (with Gen2-IB)
    - Enhanced scalability for RDMA-based direct one-sided communication with less communication resource
    - (NEW) Enhancement to the design of Win_complete for RMA operations
    - (NEW) Flexibility to support variable number of RMA windows

- Two modes of communication progress
    - Polling

  - Blocking (enables running multiple MPI processes/processor). Available for Open Fabrics Gen2-IB interface.

- Scalable job startup schemes
  - Enhanced and robust mpirun_rsh framework
  - Hierarchical ssh-based schemes to nodes
  - (NEW) Flexibility for process execution with alternate group IDs
  - Using in-band IB communication with MPD
  - Ring-based startup for RoCE
  - Support for SLURM

- Advanced AVL tree-based Resource-aware registration cache

- Memory Hook Support provided by integration with `ptmalloc2` library. This provides safe release of memory to the Operating System and is expected to benefit the memory usage of applications that heavily use `malloc` and `free` operations.

- High Performance and Portable Support for multiple networks and operating systems through uDAPL interface.
  - InfiniBand (tested with)
    * uDAPL over OpenFabrics Gen2-IB on Linux
    * uDAPL over IBTL on Solaris

  This uDAPL support is generic and can work with other networks that provide uDAPL interface. Please note that the stability and performance of MVAPICH2 with uDAPL depends on the stability and performance of the uDAPL library used. Starting from version 1.2, MVAPICH2 supports both uDAPL v2 and v1.2 on Linux.

- Support for TotalView debugger with mpirun_rsh framework.

- Shared Library Support for existing binary MPI application programs to run

- ROMIO Support for MPI-IO
  - Optimized, high-performance ADIO driver for Lustre

- Single code base for the following platforms (Architecture, OS, Compilers, Devices and InfiniBand adapters)
  - Architecture: (tested with) EM64T, Opteron and IA-32; IBM PPC and Mac G5
  - Operating Systems: (tested with) Linux and Solaris; and Mac OSX
  - Compilers: (tested with) gcc, intel, pathscale, pgi and sun studio
  - Devices: (tested with) OpenFabrics Gen2-IB, OpenFabrics Gen2-iWARP, and uDAPL; and TCP/IP
  - InfiniBand adapters (tested with):

        - Mellanox InfiniHost adapters (SDR and DDR)
        - Mellanox ConnectX (DDR and QDR with PCIe2)
        - Mellanox ConnectX-2 (QDR with PCIe2
        - QLogic adapter (SDR)
        - QLogic adapter (DDR with PCIe2
    - 10GigE adapters:
        - (tested with) Chelsio T3 adapter with iWARP support
        - (tested with) Mellanox ConnectX-EN adapter (DDR)
        - (NEW) (tested with) Intel NE020 adapter with iWARP support

The MVAPICH2 1.5 package and the project also includes the following provisions:

- Public SVN access of the codebase
- A set of micro-benchmarks (including multi-threading latency test) for carrying out MPI-level performance evaluation after the installation
- Public mvapich-discuss mailing list for mvapich users to
    - Ask for help and support from each other and get prompt response
    - Enable users and developers to contribute patches and enhancements

4 Installation Instructions

The MVAPICH2 installation process is designed to enable the most widely utilized features on the target build OS by default. Supported operating systems include Linux and Solaris. The default interface is OFA-IB-CH3/OFA-IWARP-CH3 on Linux and uDAPL on Solaris. The other interfaces, as indicated in Figure 1, can also be selected on Linux. The installation section provides generic instructions for building from a tarball or our latest sources. Please see the subsection for the interface you are targeting for specific configuration instructions.

4.1 Building from a tarball

The MVAPICH2 1.5 source code package includes MPICH2 1.2.1p1. All the required files are present as a single tarball. Download the most recent distribution tarball from: http://mvapich.cse.ohio-state.edu/download/mvapich2

Unpack the tarball and use the standard GNU procedure to compile:

```
$ tar -xzf mvapich2-1.5.tgz
$ cd mvapich2-1.5
$ ./configure
$ make
$ make install
```

In order to install a debug build, please use the following configuration option. *Please note that using debug builds may impact performance.*

```
$ ./configure --enable-g=all --enable-error-messages=all
$ make
$ make install
```

4.2 Obtaining and Building the Source from SVN repository

These instructions assume you have already installed subversion.

The MVAPICH2 SVN repository is available at:

https://mvapich.cse.ohio-state.edu/svn/mpi/mvapich2

Please keep in mind the following guidelines before deciding which version to check out:

- "tags/1.5" is the exact version released with no updates for bug fixes or new features.
  - To obtain the source code from tags/1.5:
    ```
    $ svn co https://mvapich.cse.ohio-state.edu/svn/mpi/mvapich2/tags/1.5
    mvapich2
    ```
- "branches/1.5" is a stable version with bug fixes. New features are not added to this branch.

  - To obtain the source code from branches/1.5:
    `$ svn co https://mvapich.cse.ohio-state.edu/svn/mpi/mvapich2/branches/1.5 mvapich2`

- "trunk" will contain the latest source code as we enhance and improve MVAPICH2. It may contain newer features and bug fixes, but is lightly tested.

  - To obtain the source code from trunk:
    `$ svn co https://mvapich.cse.ohio-state.edu/svn/mpi/mvapich2/trunk mvapich2`

The mvapich2 directory under your present working directory contains a working copy of the MVAPICH2 source code. Now that you have obtained a copy of the source code, you need to update the files in the source tree:

```
$ cd mvapich2
$ maint/updatefiles
```

This script will generate all of the source and configuration files you need to build MVAPICH2. If the command "autoconf" on your machine does not run autoconf 2.63 or later, but you do have a new enough autoconf available, then you can specify the correct one with the AUTOCONF environment variable (the AUTOHEADER environment variable is similar). Once you've prepared the working copy by running maint/updatefiles, just follow the usual configuration and build procedure:

```
$ ./configure
$ make
$ make install
```

4.3 Selecting a Process Manager

For the past several releases of MVAPICH2 (including this one), the mpirun_rsh/mpispawn framework from the MVAPICH distribution is now provided as an alternative to to mpd/mpiexec. By default both process managers are installed.

The mpirun_rsh/mpispawn framework launches jobs on demand in a manner more scalable than mpd/mpiexec. Using mpirun_rsh also alleviates the need to start daemons in advance on nodes used for MPI jobs. Please see Sec. 5.2.1 for more details.

This release of MVAPICH2 also supports the Hydra process manager (see Sec. 5.2.3) of the latest MPICH2 stack.

Either mpirun_rsh or Hydra can be used with any of the eight interfaces of this MVAPICH2 release, as indicated in Figure 1.

4.3.1 Using SLURM

There is now a configuration option that can be used to allow mpicc and the other MPI compiler commands to automatically link MPI programs to the SLURM's PMI library.

```
$ ./configure --with-slurm=<path to slurm installation>
```

4.4 Configuring a build for OFA-IB-CH3/OFA-iWARP-CH3/OFA-RoCE-CH3

OpenFabrics (OFA) IB/iWARP/RoCE with the CH3 channel is the default interface on Linux. It can be explicitly selected by configuring with:

```
$ ./configure --with-rdma=gen2
```

Configuration Options for OpenFabrics IB/iWARP/RoCE

- Path to OpenFabrics Header Files
  - Default: Your PATH
  - Specify: `--with-ib-include=path`
- Path to OpenFabrics Libraries
  - Default: The systems search path for libraries.
  - Specify: `--with-ib-libpath=path`
- Support for RDMA CM
  - Default: enabled, except when BLCR support is enabled
  - Disable: `--disable-rdma-cm`
- Support for RoCE
  - Default: enabled
  - For RoCE functionality to work properly, a version of OFED from the OFED-1.5-RoCE branch must be installed on all the systems.
- Registration Cache
  - Default: enabled
  - Disable: `--disable-registration-cache`
- ADIO driver for Lustre:
  - When compiled with this support, MVAPICH2 will use the optimized driver for Lustre. In order to enable this feature, the flag
`--enable-romio --with-file-system=lustre`
should be passed to `configure` (`--enable-romio` is optional as it is enabled by default). You can add support for more file systems using
`--enable-romio --with-file-system=lustre+nfs+pvfs2`
- LiMIC2 Support
  - Default: disabled

  - Enable:
    `--with-limic2[=<path to LiMIC2 installation>]`
    `--with-limic2-include=<path to LiMIC2 headers>`
    `--with-limic2-libpath=<path to LiMIC2 library>`

- Header Caching
  - Default: enabled
  - Disable: `--disable-header-caching`

- Berkeley Lab Checkpoint/Restart Support
  - Default: disabled
  - Enable: `--enable-blcr`
    `--with-blcr-libpath=path --with-blcr-include=path`

- Berkeley Lab Checkpoint/Restart Support with FTB
  - Default: disabled
  - Enable: `--enable-blcr --enable-ftb`
    `--with-blcr-libpath=path --with-blcr-include=path`
    `--with-ftb-libpath=path --with-ftb-include=path`

- eXtended Reliable Connection
  - Default: disabled
  - Disable: `--enable-xrc`

- HWLOC Support (Affinity)
  - Default: enabled
  - Disable: `--without-hwloc`

4.5 Configuring a build for OFA-IB-Nemesis

The Nemesis sub-channel now is supported over OFA-IB. It can be built with:

```
$ ./configure --with-device=ch3:nemesis:ib
```

Configuration options for OFA-IB-Nemesis:

- Path to OpenFabrics Header Files
  - Default: Your PATH
  - Specify: `--with-ib-include=path`

- Path to OpenFabrics Libraries
  - Default: Your PATH

  - Specify: `--with-ib-libpath=path`

- Registration Cache
  - Default: enabled
  - Disable: `--disable-registration-cache`

- Header Caching
  - Default: enabled
  - Disable: `--disable-header-caching`

4.6 Configuring a build for uDAPL-CH3

The uDAPL interface is the default on Solaris. It can be explicitly selected on both Solaris and Linux by configuring with:

```
$ ./configure --with-rdma=udapl
```

Configuration options for uDAPL

- Path to OpenFabrics Header Files
  - Default: Your PATH
  - Specify: `--with-ib-include=path`

- Path to OpenFabrics Libraries
  - Default: The systems search path for libraries.
  - Specify: `--with-ib-libpath=path`

- Path to the DAPL Header Files
  - Default: Your PATH
  - Specify: `--with-dapl-include=path`

- Path to the DAPL Library
  - Default: The systems search path for libraries.
  - Specify: `--with-dapl-libpath=path`

- Default DAPL Provider
  - Default: OpenIB-cma on Linux; ibd0 on Solaris
  - Specify: `--with-dapl-provider=type`
    - Where `type` can be found in:
      - /etc/dat.conf on Linux
      - /etc/dat/dat.conf on Solaris

- DAPL Version
  - Default: 1.2
  - Specify: `--with-dapl-version=version`
- Cluster Size
  - Default: small
  - Specify: `--with-cluster-size=level`
    - Where `level` is one of:
      - small: < 128 processor cores
      - medium: 128 - 1024 cores
      - large: > 1024 cores
- I/O Bus
  - Default: PCI Express
  - Specify: `--with-io-bus=type`
    - Where `type` is one of:
      - PCI_EX for PCI Express
      - PCI_X for PCI-X
- Link Speed
  - Default: SDR
  - Specify: `--with-link=type`
    - Where `type` is one of:
      - DDR
      - SDR
- Registration Cache
  - Default: enabled on Linux; enabled and not configurable on Solaris
  - Disable (Linux only): `--disable-registration-cache`
- Header Caching
  - Default: enabled
  - Disable: `--disable-header-caching`
- HWLOC Support (Affinity)
  - Default: enabled
  - Disable: `--without-hwloc`

4.7 Configuring a build for QLogic PSM-CH3

The QLogic PSM interface needs to be built to use MVAPICH2 on InfiniPath adapters. It can built with:

```
$ ./configure --with-device=ch3:psm
```

Configuration options for QLogic PSM channel

- Path to QLogic PSM header files
  - Default: The systems search path for header files
  - Specify: `--with-psm-include=path`
- Path to QLogic PSM library
  - Default: The systems search path for libraries
  - Specify: `--with-psm=path`

To build and install the library we will need to run:

```
$ make
```

```
$ make install
```

4.8 Configuring a build for TCP/IP-CH3

The use of TCP/IP requires the explicit selection of a TCP/IP enabled channel. The recommended channel is ch3:sock and it can be selected by configuring with:

```
$ ./configure --with-device=ch3:sock
```

Additional instructions for configuring with TCP/IP can be found in the MPICH2 documentation available at:

http://www.mcs.anl.gov/research/projects/mpich2/documentation/index.php?s=docs

4.9 Configuring a build for TCP/IP-Nemesis

The use of TCP/IP with Nemesis channel requires the following configuration:

```
$ ./configure --with-device=ch3:nemesis
```

Additional instructions for configuring with TCP/IP-Nemesis can be found in the MPICH2 documentation available at: `http://www.mcs.anl.gov/research/projects/mpich2/documentation/index.php?s=docs`

5 Basic Usage Instructions

5.1 Compile Applications

MVAPICH2 provides a variety of MPI compilers to support applications written in different programming languages. Please use `mpicc, mpif77, mpiCC`, or `mpif90` to compile applications. The correct compiler should be selected depending upon the programming language of your MPI application.

These compilers are available in the `MVAPICH2_HOME/bin` directory. MVAPICH2 installation directory can also be specified by modifying `$PREFIX`, then all the above compilers will also be present in the `$PREFIX/bin` directory.

5.2 Run Applications

This section provides instructions on how to run applications with MVAPICH2. Please note that on new multi-core architectures, process-to-core placement has an impact on performance. Please refer to Section 6.2 to learn about running MVAPICH2 library on multi-core nodes.

5.2.1 Running using `mpirun_rsh` (for all interfaces including OFA-IB-CH3, OFA-IB-Nemesis, OFA-iWARP-CH3, OFA-RoCE-CH3, PSM-CH3, uDAPL-CH3, TCP/IP-CH3 and TCP/IP-Nemesis)

The MVAPICH team suggests users using this mode of job start-up for all interfaces. This `mpirun_rsh` scheme provides fast and scalable job start-up. It scales to multi-thousand node clusters.

Prerequisites:

- Either `ssh` or `rsh` should be enabled between the front nodes and the computing nodes. In addition to this setup, you should be able to login to the remote nodes without any password prompts.
- All hostnames should resolve to the same IP address on all machines. For instance, if a machine's hostnames resolves to 127.0.0.1 due to the default /etc/hosts on some Linux distributions it leads to incorrect behavior of the library.

Examples of running programs using `mpirun_rsh`:

```
$ mpirun_rsh -np 4 n0 n1 n2 n3 ./cpi
```

This command launches `cpi` on nodes n0, n1, n2 and n3, one process per node. By default `ssh` is used.

```
$ mpirun_rsh -rsh -np 4 n0 n1 n2 n3 ./cpi
```

This command launches `cpi` on nodes n0, n1, n2 and n3, one process per each node using `rsh` instead of `ssh`.

```
$ mpirun_rsh -np 4 -hostfile hosts ./cpi
```

A list of target nodes must be provided in the file `hosts` one per line. MPI ranks are assigned in order of the hosts listed in the hosts file or in the order they are passed to mpirun_rsh. i.e., if the nodes are listed as n0 n1 n0 n1, then n0 will have two processes, rank 0 and rank 2; whereas n1 will have rank 1 and 3. This rank distribution is known as "cyclic". If the nodes are listed as n0 n0 n1 n1, then n0 will have ranks 0 and 1; whereas n1 will have ranks 2 and 3. This rank distribution is known as "block".

Many parameters of the MPI library can be configured at run-time using environmental variables. In order to pass any environment variable to the application, simply put the variable names and values just before the executable name, like in the following example:

```
$ mpirun_rsh -np 4 -hostfile hosts ENV1=value ENV2=value ./cpi
```

Note that the environmental variables should be put immediately before the executable.

Alternatively, you may also place environmental variables in your shell environment (e.g. `.bashrc`). These will be automatically picked up when the application starts executing.

Other options of `mpirun_rsh` can be obtained using

```
$ mpirun_rsh --help
```

Note that mpirun_rsh is sensitive to the ordering of the command-line options.

There are many different parameters which could be used to improve the performance of applications depending upon their requirements from the MPI library. For a discussion on how to identify such parameters, see Section 9.

Job Launch using MPMD execution mode: The mpirun_rsh framework also supports job launching using MPMD mode. It permits the use of heterogeneous jobs using multiple executables and command line arguments. The following format needs to be used:

```
$ mpirun_rsh -config configfile -hostfile hosts
```

A list of different group of executables must be provided to the job launcher in the file `configfile`, one per line. The `configfile` can contain comments. Lines beginning with "#" are considered comments.

For example:

```
#Config file example

#Launch 4 copies of exe1 with arguments arg1 and arg2

-n 4 :  exe1 arg1 arg2

#Launch 2 copies of exe2

-n 2 :  exe2
```

A list of target nodes must be provided in the file `hosts` one per line and the allocation policy previously described is used.

Please note that this section only gives general information on how to run applications using mpirun_rsh. Please refer to the following sections for more information on how to run the application over various devices such as iWARP and RoCE.

5.2.2 Execution using an alternate group ID

mpirun_rsh can launch the tasks using an alternate group ID with the "*-sg group*" option. For example:

```
mpirun_rsh -sg secondarygroup -np 2 host1 host2 ./a.out
```

This command executes the program *a.out* running the remote tasks using the *secondarygroup* as their group ID.

5.2.3 Running using `mpiexec.hydra` (for all interfaces including OFA-IB-CH3, OFA-IB-Nemesis, OFA-iWARP-CH3, OFA-RoCE-CH3, PSM-CH3, uDAPL-CH3, TCP/IP-CH3 and TCP/IP-Nemesis)

The `mpiexec.hydra` is a process management system from Argonne National Laboratory (http://wiki.mcs.anl.gov/mpich2/index.php/Using_the_Hydra_Process_Manager). The following is an examples of running programs using `mpiexec.hydra`:

```
mpiexec.hydra -f hosts -n 2 ./cpi
```

This command executes the *cpi* program on two hosts, whose names are in the file "hosts".

This process manager has many features. Please refer to the above-mentioned Web page for more details.

5.2.4 Running using SLURM

SLURM is an open-source resource manager designed by Lawrence Livermore National Laboratory. SLURM software package and its related documents can be downloaded from: http://www.llnl.gov/linux/slurm

Once SLURM is installed and the daemons are started, applications compiled with MVAPICH2 can be launched by SLURM, e.g.

```
$ srun -n2 --mpi=none ./a.out
```

The use of SLURM enables many good features such as explicit CPU and memory binding. For example, if you have two processes and want to bind the first process to CPU 0 and Memory 0, and the second process to CPU 4 and Memory 1, then it can be achieved by:

```
$ srun --cpu_bind=v,map_cpu:0,4 --mem_bind=v,map_mem:0,1
-n2 --mpi=none ./a.out
```

For more information about SLURM and its features please visit SLURM website.

5.2.5 Running with Dynamic Process Management support

MVAPICH2 (OFA-IB-CH3 interface) provides MPI-2 dynamic process management. This feature allows MPI applications to spawn new MPI processes according to MPI-2 semantics. The following commands provide an example of how to run your application.

- To run your application using mpirun_rsh
  ```
  $ mpirun_rsh -np 2 -hostfile hosts MV2_SUPPORT_DPM=1 ./spawn1
  ```
  Note: It is necessary to provide the hostfile when running dynamic process management applications using mpirun_rsh.
- To run your application using mpiexec.hydra
  ```
  $ mpiexec.hydra -n 2 -env MV2_SUPPORT_DPM 1 ./spawn1
  ```

Please refer to Section 11.62 for information about the MV2_SUPPORT_DPM environment variable.

5.2.6 Running with mpirun_rsh using OFA-iWARP Device

In MVAPICH2, OFA-iWARP support is enabled with the use of the run time environment variable `MV2_USE_IWARP_MODE`.

In addition to this flag, all the systems to be used need the following one time setup for enabling RDMA CM usage.

- **Setup the RDMA CM device:** RDMA CM device needs to be setup, configured with an IP address and connected to the network.
- **Setup the Local Address File:** Create the file (`/etc/mv2.conf`) with the local IP address to be used by RDMA CM. (Multiple IP addresses can be listed (one per line) for multirail configurations).
  ```
  $ echo 10.1.1.1 >> /etc/mv2.conf
  ```

Programs can be executed as follows:

```
$ mpirun_rsh -np 2 MV2_USE_IWARP_MODE=1 ENV1=value1 prog
```

The iWARP device also provides TotalView debugging and shared library support. Please refer to section 5.2.11 and 5.2.12 for shared library and TotalView support, respectively.

5.2.7 Running with mpirun_rsh using OFA-RoCE Device

In MVAPICH2, OFA-RoCE support is enabled with the use of the run time environment variable `MV2_USE_RDMAOE`.

Programs can be executed as follows:

```
$ mpirun_rsh -np 2 MV2_USE_RDMAOE=1 ENV1=value1 prog
```

5.2.8 Running using mpiexec.hydra with uDAPL-CH3 Device

MVAPICH2 can be configured with the uDAPL device, as described in the Section 4.6 . To compile MPI applications, please refer to the Section 5.1. In order to run MPI applications with uDAPL support, please specify the environmental variable `MV2_DAPL_PROVIDER`. As an example,

```
$ mpiexec.hydra -n 4 -env MV2_DAPL_PROVIDER OpenIB-cma ./cpi
```

or:

```
$ export MV2_DAPL_PROVIDER=OpenIB-cma
$ mpiexec.hydra -n 4 ./cpi
```

Please check the `/etc/dat.conf` file on Linux or `/etc/dat/dat.conf` on Solaris to find all the available uDAPL service providers. The default value for the uDAPL provider will be chosen, if no environment variable is provided at runtime. If you are using OpenFabrics software stack on Linux, the default DAPL provider is `OpenIB-cma` for DAPL-1.2, and `ofa-v2-ib0` for DAPL-2.0. If you are using Solaris, the default DAPL provider is `ibd0`.

The uDAPL device also provides TotalView debugging and shared library support. Please refer to section 5.2.11 and 5.2.12 for shared library and TotalView support, respectively.

5.2.9 Running using mpiexec.hydra with TCP/IP

You would like to run an MPI job using IPoIB but your IB card is not the default interface for IP traffic. Assume that you have a cluster setup as the following:

#hostname	Eth Addr	IPoIB Addr
compute1	192.168.0.1	192.168.1.1
compute2	192.168.0.2	192.168.1.1
compute3	192.168.0.3	192.168.1.1
compute4	192.168.0.4	192.168.1.1

You will need to create a machine file for mpiexec.hydra that tells it to use a particular interface, and then use the *-iface* option. Example:

```
$ cat - > $(MACHINE_FILE) compute1
compute2
compute3
compute4

$ mpiexec.hydra -f MACHINE_FILE -iface ib0 -n 4 ./app1
```

More information is at the url http://wiki.mcs.anl.gov/mpich2/index.php/Using_the_Hydra_Process_Manager#Hydra_with_Non-Ethernet_Networks.

5.2.10 Running using ADIO driver for Lustre

MVAPICH2 contains optimized Lustre ADIO support for the OFA-IB-CH3 device. The Lustre directory should be mounted on all nodes on which MVAPICH2 processes will be running. Compile MVAPICH2 with ADIO support for Lustre as described in Section 4. If your Lustre mount is /mnt/datafs on nodes n0 and n1, on node n0, you can compile and run your program as follows:

```
$ mpicc -o perf romio/test/perf.c
$ mpirun_rsh -np 2 n0 n1 <path to perf>/perf -fname /mnt/datafs/testfile
```

If you have enabled support for multiple file systems, append the prefix "lustre:" to the name of the file. For example:

```
$ mpicc -o perf romio/test/perf.c
$ mpirun_rsh -np 2 n0 n1 ./perf -fname lustre:/mnt/datafs/testfile
```

5.2.11 Running using Shared Library Support

MVAPICH2 provides shared library support. This feature allows you to build your application on top of MPI shared library. If you choose this option, you still will be able to compile applications with static libraries. But as default, when you have shared library support enabled your applications will be built on top of shared libraries automatically. the following commands provide some examples of how to build and run your application with shared library support.

- To compile your application with shared library support. Run the following command:
  ```
  $ mpicc -o cpi cpi.c
  ```
- To execute an application compiled with shared library support, you need to specify the path to the shared library by putting LD_LIBRARY_PATH=path-to-shared-libraries in the command line. For example:
  ```
  $ mpiexec.hydra -np 2 -env LD_LIBRARY_PATH $MVAPICH2_INSTALL/lib/shared
  ./cpi
  ```
  or
  ```
  $ mpirun_rsh -np 2 n0 n1 LD_LIBRARY_PATH=/path/to/shared/lib ./cpi
  ```
  .
- To disable MVAPICH2 shared library support even if you have installed MVAPICH2, run the following command:
  ```
  $ mpicc -noshlib -o cpi cpi.c
  ```

5.2.12 Running using TotalView Debugger Support

MVAPICH2 provides TotalView support. The following commands provide an example of how to build and run your application with TotalView support. Note: running TotalView requires correct setup in your environment, if you encounter any problem with your setup, please check with your system administrator for help.

- Define ssh as a `TVDSVRLAUNCHCMD` variable in your default shell. For example, with bashrc, you can do:

```
$ echo "export TVDSVRLAUNCHCMD=ssh" >> $HOME/.bashrc
```

- Configure MVAPICH2 with the configure options `--enable-g=dbg --enable-sharedlibs=<kind> --enable-debuginfo CFLAGS='-D_XOPEN_SOURCE=600'` in addition to the default options and then build MVAPICH2. *kind* may be *gcc* for standard gcc, *osx-gcc* for OS/X and *solaris-cc* for solaris native compilers
- Compile your program with a flag -g:
  ```
  $ mpicc -g -o prog prog.c
  ```
- Define the correct path to TotalView as the TOTALVIEW variable. For example, for mpirun_rsh, under the bash shell:
  ```
  $ export TOTALVIEW=<path_to_TotalView>
  ```
  or for mpiexec, under bash shell:
  ```
  $ export MPIEXEC_TOTALVIEW=<path_to_TotalView>
  ```
- Run your program:
  ```
  $ mpirun_rsh -tv -np 2 n0 n1 LD_LIBRARY_PATH=$MVAPICH2_INSTALL/lib/shared:
  $MVAPICH2_INSTALL/lib prog
  ```
  or
  ```
  $ mpiexec -tv -np 2 -env LD_LIBRARY_PATH $MVAPICH2_INSTALL/lib/shared:
  $MVAPICH2_INSTALL/lib prog
  ```
- Troubleshooting:
  - X authentication errors: check if you have enabled X Forwarding
    ```
    $ echo "ForwardX11 yes" >> $HOME/.ssh/config
    ```
  - ssh authentication error: ssh to the computer node with its long form hostname, for example, ssh i0.domain.osu.edu

6 Advanced Usage Instructions

In this section, we present the usage instructions for advanced features provided by MVAPICH2.

6.1 Running with Customized Optimizations (for OFA-IB-CH3, OFA-iWARP-CH3 and OFA-RoCE-CH3 Devices)

In MVAPICH2 1.5 , run-time variables are used to switch various optimization schemes on and off. Following is a list of optimizations schemes and the control environmental variables, for a full list please refer to the section 11:

- **Adaptive RDMA fast path:** using RDMA write to enhance performance for short messages. Default: on; to disable:
  `$ mpirun_rsh -np 2 n0 n1 MV2_USE_RDMA_FAST_PATH=0 prog`
  or
  `$ mpiexec.hydra -n 2 -env MV2_USE_RDMA_FAST_PATH 0 prog`

- **Shared-receive Queue:** This feature is available only with Gen2-IB devices. This is targeted for using Shared Receive Queue (SRQ). Default: on; to disable:
  `$ mpirun_rsh -np 2 n0 n1 MV2_USE_SRQ=0 prog`
  or
  `$ mpiexec.hydra -n 2 -env MV2_USE_SRQ 0 prog`

- **Optimizations for one sided communication:** One sided operations can be directly built on RDMA operations. Currently this scheme will be disabled if on-demand connection management is used. Default: on; to disable:
  `$ mpirun_rsh -np 2 n0 n1 MV2_USE_RDMA_ONE_SIDED=0 prog`
  or
  `$ mpiexec.hydra -n 2 -env MV2_USE_RDMA_ONE_SIDED 0 prog`

- **Lazy memory unregistration:** user-level registration cache. Default: on; to disable:
  `$ mpirun_rsh -np 2 n0 n1 MV2_USE_LAZY_MEM_UNREGISTER=0 prog`
  or
  `$ mpiexec.hydra -n 2 -env MV2_USE_LAZY_MEM_UNREGISTER 0 prog`

6.2 Running with Efficient CPU (Core) Mapping

MVAPICH2-CH3 interfaces support architecture specific CPU mapping through the Portable Hardware Locality (hwloc) software package. By default, the HWLOC sources are compiled and built while the MVAPICH2 library is being installed. Users can choose the “–disable-hwloc” parameter while configuring the library if they do not wish to have the HWLOC library installed. However, in such cases, the MVAPICH2 library will not be able to perform any affinity related operations.

There are two major schemes as indicated below. To take advantage of any of these schemes, the jobs need to run with CPU affinity turned on (default). If users choose to set MV2_ENABLE_AFFINITY to 0, then the kernel takes care of mapping processes to cores and none of these schemes will be enabled.

6.2.1 Using HWLOC for CPU Mapping

Under this scheme, the HWLOC tool will be used at job-launch time to detect the processor's micro-architecture, and then generate a suitable cpu mapping string based. Two policies are currently implemented: "bunch" and "scatter". By default, we choose to use the "bunch" mapping. However, we also allow users to choose a binding policy through the run-time variable, MV2_CPU_BINDING_POLICY. (Section 11.11)

For example, if you want to run 4 processes per node and utilize "bunch" policy on each node, you can specify:

```
$mpirun_rsh -np 4 -hostfile hosts MV2_CPU_BINDING_POLICY=bunch ./a.out
```

The CPU binding will be set as shown in Figure 2.

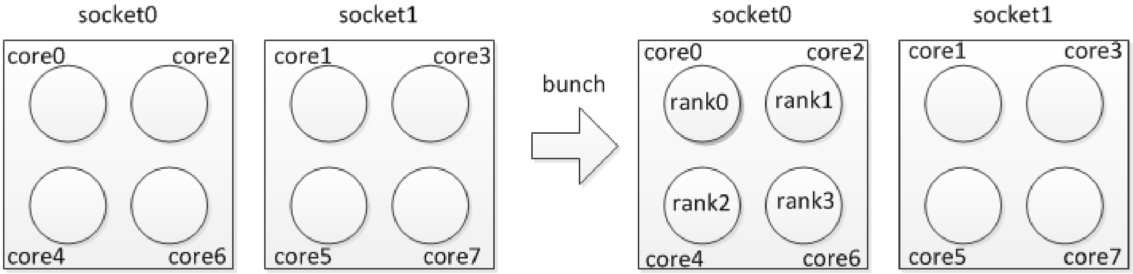

Figure 2: Process placement with "bunch" CPU binding policy

If you want to run 4 processes per node and utilize "scatter" policy on each node, you can specify:

```
$mpirun_rsh -np 4 -hostfile hosts MV2_CPU_BINDING_POLICY=scatter ./a.out
```

The CPU binding will be set as shown in Figure 3.

If two applications with four processes each need to share a given node (with eight cores) at the same time with "bunch" policy, you can specify:

```
$mpirun_rsh -np 4 -hostfile hosts MV2_CPU_BINDING_POLICY=bunch ./a.out
```

```
$mpirun_rsh -np 4 -hostfile hosts MV2_CPU_BINDING_POLICY=bunch ./b.out
```

The CPU binding will be set as shown in Figure 4.

If two applications with four processes each need to share a given node (with eight cores) at the

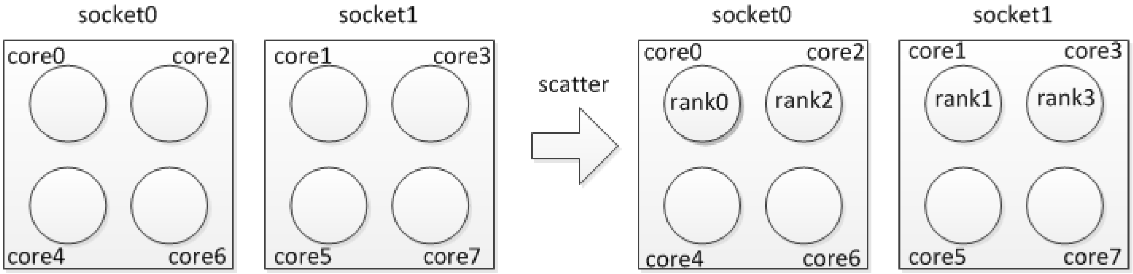

Figure 3: Process placement with "Scatter" CPU binding policy

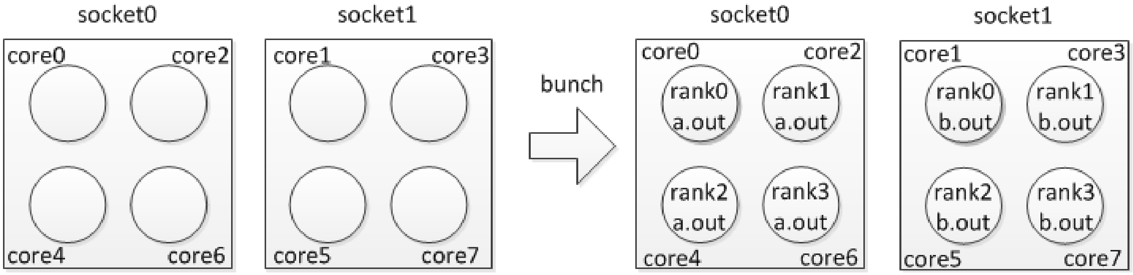

Figure 4: Process placement with two applications using the "bunch" CPU binding policy

same time with "scatter" policy, you can specify:

```
$mpirun_rsh -np 4 -hostfile hosts MV2_CPU_BINDING_POLICY=scatter ./a.out
$mpirun_rsh -np 4 -hostfile hosts MV2_CPU_BINDING_POLICY=scatter ./b.out
```

The CPU binding will be set as shown in Figure 5.

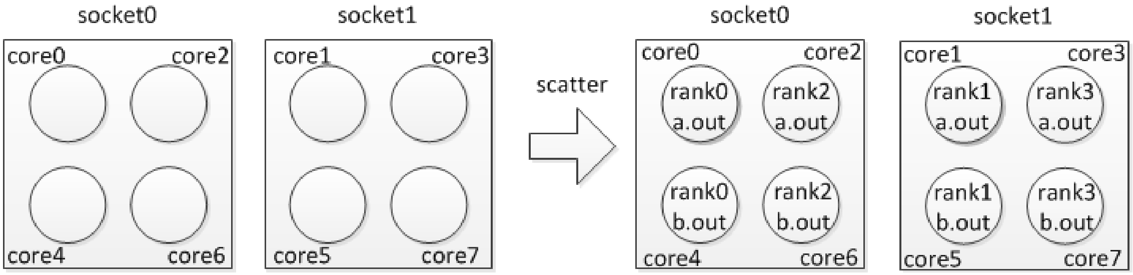

Figure 5: Process placement with two applications using the "scatter" CPU binding policy

6.2.2 User defined CPU Mapping

Under the second scheme, users can also use their own mapping to bind processes to CPU's on modern multi-core systems. The feature is especially useful on multi-core systems, where performance may be different if processes are mapped to different cores. The mapping can be specified by setting the environment variable `MV2_CPU_MAPPING` (Section 11.10).

For example, if you want to run 4 processes per node and utilize cores 0, 1, 4, 5 on each node, you can specify:

```
$ mpirun_rsh -np 64 -hostfile hosts MV2_CPU_MAPPING=0:1:4:5 ./a.out
```

or

```
$ mpiexec.hydra -n 64 -env MV2_CPU_MAPPING 0:1:4:5 ./a.out
```

In this way, process 0 on each node will be mapped to core 0, process 1 will be mapped to core 1, process 2 will be mapped to core 4, and process 3 will be mapped to core 5. For each process, the mapping is separated by a single ":".

6.2.3 Performance Impact of CPU Mapping

Here we provide a table with latency performance of 0 byte and 8KB messages using different CPU mapping schemes. The results show how process binding can affect the benchmark performance. We strongly suggest the consideration of best CPU mapping on multicore platforms when carrying out benchmarking and performance evaluation with MVAPICH2.

The following measurements were taken on the machine with the dual quad-core Intel Xeon E5530 2.40GHz processors with 8MB L3 shared cache. MVAPICH2-1.5 was built with gcc-4.1.2 and default configure arguments:

	Message Latency		
Core Pair	0-byte	8k-byte	Notes
2,4	0.26 us	1.98 us	same socket, shared L3 cache, best performance
0,2	0.27 us	2.09 us	same socket, shared L3 cache, but core 0 handles interrupts
2,3	0.67 us	3.40 us	different sockets
0,1	0.71 us	3.64 us	different sockets, but core 0 handles interrputs

6.3 Running with LiMIC2

MVAPICH2 CH3-based interfaces (OFA-IB-CH3, OFA-iWARP-CH3 and OFA-RoCE-CH3) support LiMIC2 for intra-node communication for medium and large messages to get higher performance. It is disabled by default because it depends on the LiMIC2 package to be previously installed. As a convenience we have distributed the latest LiMIC2 package (as of this release) with our sources.

To install this package, please take the following steps.

- Navigate to the LiMIC2 to source
  ```
  $ cd limic2-0.5.3
  ```
- Configure and build the source
  ```
  limic2-0.5.3$ ./configure --enable-module --sysconfdir=/etc && make
  ```
- Install
  ```
  limic2-0.5.3$ sudo make install
  ```

Before using LiMIC2 you'll need to load the kernel module. If you followed the instructions above you can do this using the following command (LSB init script).

- `$ /etc/init.d/limic start`

Please note that supplying '`--sysconfdir=/etc`' in the configure line above told the package to install the init script and an udev rule in the standard location for system packages. This is optional but recommended.

Now you can use LiMIC2 with MVAPICH2 by simply supplying the '`--with-limic2`' option when configuring MVAPICH2. You can run your applications as normal and LiMIC2 will be used for medium and large intra-node messages. To disable it at run time, use the env variable:

```
$ mpirun_rsh -np 64 -hostfile hosts MV2_SMP_USE_LIMIC2=0 ./a.out
```

6.4 Running with Shared Memory Collectives

In MVAPICH2, support for shared memory based collectives has been enabled for MPI applications running over OFA-IB-CH3, OFA-iWARP-CH3 and uDAPL-CH3 stack. Currently, this support is available for the following collective operations:

- MPI_Allreduce
- MPI_Reduce
- MPI_Barrier
- MPI_Bcast

Optionally, these feature can be turned off at runtime by using the following parameters:

- `MV2_USE_SHMEM_COLL` (section 11.80)
- `MV2_USE_SHMEM_ALLREDUCE` (section 11.77)
- `MV2_USE_SHMEM_REDUCE` (section 11.81)
- `MV2_USE_SHMEM_BARRIER` (section 11.78)
- `MV2_USE_SHMEM_BCAST` (section 11.79)

Please refer to Section 11 for further details.

6.5 Running with Multiple-Rail Configurations (for OFA-IB-CH3 and OFA-iWARP-CH3 Devices)

MVAPICH2 has integrated multi-rail support. Run-time variables are used to specify the control parameters of the multi-rail support; number of adapters with MV2_NUM_HCAS (section 11.35), number of ports per adapter with MV2_NUM_PORTS (section 11.36), and number of queue pairs per port with MV2_NUM_QP_PER_PORT (section 11.37). Those variables are default to 1 if you do not specify them.

Large messages are striped across all HCA's. The threshold for striping = (MV2_VBUF_TOTAL_SIZE × MV2_NUM_PORTS × MV2_NUM_QP_PER_PORT × MV2_NUM_HCAS).

MVAPICH2 also gives the flexibility to balance short message traffic over multiple HCAs in a multi-rail configuration. The run-time variable MV2_SM_SCHEDULING can be used to choose between the various load balancing options available. It can be set to USE_FIRST (Default) or ROUND_ROBIN. In the USE_FIRST scheme, the HCA in slot 0 is always used to transmit the short messages. If ROUND_ROBIN is chosen, messages are sent across all HCAs alternately.

Following is an example to run multi-rail support with two adapters, using one port per adapter and one queue pair per port:

```
$ mpirun_rsh -np 2 n0 n1 MV2_NUM_HCAS=2 MV2_NUM_PORTS=1
MV2_NUM_QP_PER_PORT=1 prog
or
$ mpiexec.hydra -n 2 -env MV2_NUM_HCAS 2 -env MV2_NUM_PORTS 1 -env
MV2_NUM_QP_PER_PORT 1 prog
```

Note that you don't need to specify `MV2_NUM_PORTS` and `MV2_NUM_QP_PER_PORT` since they default to 1, so you can type:

```
$ mpirun_rsh -np 2 n0 n1 MV2_NUM_HCAS=2 prog
or
$ mpirun_rsh -np 2 n0 n1 MV2_NUM_HCAS=2 MV2_SM_SCHEDULING=ROUND_ROBIN prog
or
$ mpiexec.hydra -n 2 -env MV2_NUM_HCAS 2 prog
```

The user can also select the the particular network card that should be used by using the MV2_IBA_HCA environment variable specified in section 11.22. The following is an example of how to run MVAPICH2 in this mode. (In the example 'mlx4_0' is the name of the InfiniBand card as displayed by the output of the 'ibstat' command).

```
$ mpirun_rsh -np 2 n0 n1 MV2_IBA_HCA=mlx4_0 prog
```

6.6 Running with Checkpoint/Restart Support (for OFA-IB-CH3 Device)

MVAPICH2 provides system-level checkpoint/restart functionality for the OpenFabrics Gen2-IB interface with the option of using BLCR in stand alone mode or using BLCR in conjunction with FTB support. FTB enables faults to be handled in a co-ordinated and holistic manner in the entire system, providing for an infrastructure which can be used by different software systems to exchange fault-related information.

Three methods are provided to invoke checkpointing: Manual, Automated and Application Initiated Synchronous Checkpointing. In order to utilize the checkpoint/restart functionality there are few steps that need to be followed.

- Download and install the BLCR (Berkeley Lab's Checkpoint/Restart) package. The packages can be downloaded from this webpage.
- Make sure the BLCR packages are installed on every node and the `LD_LIBRARY_PATH` must contain the path to the shared library of BLCR, usually `$BLCR_HOME/lib`.
- MVAPICH2 needs to be compiled with checkpoint/restart support, see section 4.4.
- BLCR's kernel modules must be loaded on all the compute nodes.
- Make sure the PATH contains the path to the executables of BLCR, usually `$BLCR_HOME/bin`.

Users are strongly encouraged to read the Administrators guide of BLCR, and test the BLCR on the target platform, before using the checkpointing feature of MVAPICH2.

If using the FTB frame-work for checkpoint/restart, in addition to the above following needs to be done.

- Download and install the FTB (Fault Tolerance Backplane) package. The packages can be downloaded from here.
- Make sure the FTB packages are installed on every node and the `LD_LIBRARY_PATH` must contain the path to the shared library of FTB, usually `$FTB_HOME/lib`.
- MVAPICH2 needs to be compiled with checkpoint/restart as well as FTB support, see section 4.4.
- Start FTB Database server (`$FTB_HOME/sbin/ftb_database_server)` on one of the nodes, this node will act as server node for all the FTB agents.
- Start FTB agents (`$FTB_HOME/sbin/ftb_agent`) on all the compute nodes.

Now, your system is set up to use the Checkpoint/Restart features of MVAPICH2. Several parameters are provided by MVAPICH2 for flexibility in configuration and using the Checkpoint / Restart features. If mpiexec is used as the job startup mechanism, these parameters need to be set in the user's environment through the BASH shell's `export` command, or the equivalent command for other shells. If mpirun_rsh is used as the job startup mechanism, these parameters need to be passed to mpirun_rsh through the command line.

- MV2_CKPT_FILE: This parameter specifies the path and the base filename for checkpoint files of MPI processes. Please note that File System performance is critical to the performance of checkpointing. This parameter controls which file system will be used to store the checkpoint files. For example, if your PVFS2 is mounted at /mnt/pvfs2, using MV2_CKPT_FILE=/mnt/pvfs2/ckptfile will let the checkpoint files being stored in pvfs2 file system. See Section 11.1 for details.

- MV2_CKPT_INTERVAL: This parameter (in minutes) can be used to enable automatic checkpointing. See Section 11.2 for details.
- MV2_CKPT_MAX_SAVE_CKPTS: This parameter is used to limit the number of checkpoints saved on file system. See Section 11.3 for details.
- MV2_CKPT_NO_SYNC: This parameter is used to control whether the program forces the checkpoint files being synced to disk or not before it continues execution. See Section 11.6 for details.
- MV2_CKPT_MPD_BASE_PORT: Not applicable to mpirun_rsh. See Section 11.4 for details.
- MV2_CKPT_MPIEXEC_PORT: Not applicable to mpirun_rsh. See Section 11.4 for details.

In order to provide maximum flexibility to end users who wish to use the checkpoint/restart features of MVAPICH2, we've provided three different methods which can be used to take the checkpoints during the execution of the MPI application. These methods are described as follows:

- Manual Checkpointing: In this mode, the user simply launches an MPI application and chooses when to checkpoint the application. This mode can be primarily used for experimentation during deployment stages. In order to use this mode, the MPI application is launched normally using mpiexec or mpirun_rsh. When the user decides to take a checkpoint, the users can issue a BLCR command called ``cr_checkpoint" with the process id (PID) of the mpiexec or mpirun_rsh process. In order to simplify the process, the script mv2_checkpoint can be used. This script is available in the same directory as mpiexec and mpirun_rsh.
- Automated Checkpointing: In this mode, the user can launch the MPI application normally using mpiexec or mpirun_rsh. However, instead of manually issuing checkpoints as described in the above bullet, a parameter (MV2_CKPT_INTERVAL) can be set to automatically take checkpoints and user-defined intervals. Please refer to Section 11.2 for a complete usage description of this variable. This mode can be used to take checkpoints of a long running application, for example every 1 hour, 2 hours etc. based on user's choice.
- Application Initiated Synchronous Checkpointing: In this mode, the MPI application which is running can itself request for a checkpoint. Application can request a whole program checkpoint synchronously by calling MVAPICH2_Sync_Checkpoint. Note that it is a collective operation, and this function must be called from all processes to take the checkpoint. This mode is expected to be used by applications that can be modified and have well defined synchronization points. These points can be effectively used to take checkpoints. An example of how this mode can be activated is given below.

```
#include "mpi.h"
```

```
#include <unistd.h>
#include <stdio.h>

int main(int argc,char *argv[])
{
    MPI_Init(&argc,&argv);
    printf("Computation\n");
    sleep(5);
    MPI_Barrier(MPI_COMM_WORLD);
    MVAPICH2_Sync_Checkpoint();
    MPI_Barrier(MPI_COMM_WORLD);
    printf("Computation\n");
    sleep(5);
    MPI_Finalize();
    return 0;
}
```

To restart a job from a checkpoint, users need to issue another command of BLCR, "`cr_restart`" with the checkpoint file name of the MPI job console as the parameter, usually `context.<pid>`. The checkpoint file name of the MPI job console can be specified when issuing the checkpoint, see the "`cr_checkpoint --help`" for more information. Please note that the names of checkpoint files of the MPI processes will be assigned according to the environment variable MV2_CKPT_FILE, (`$MV2_CKPT_FILE.<number of checkpoint>.<process rank>`).

If the user wishes to restart the MPI job on a different set of nodes, the host file that was specified along with the "-hostfile" option during job launch phase should be modified accordingly before trying to restart a job with "cr_restart". This modified "hostfile" must be at the same location and with the same file name as the original hostfile. The mpirun_rsh framework parses the host file when trying to restart from a checkpoint, and launches the job on the corresponding nodes. This is possible as long as the nodes in which the user is trying to restart has the exact same environment as the one in which the checkpoint was taken (including shared NFS mounts, kernel versions, and user libraries).

Please refer to the Section 10.6 for troubleshooting with Checkpoint/Restart.

6.7 Running Running with Network Fault Tolerance/APM Support (for OFA-IB-CH3 Device)

MVAPICH2 supports network fault recovery by using InfiniBand Automatic Path Migration mechanism. This support is available for MPI applications using OpenFabrics stack and InfiniBand adapters.

To enable this functionality, a run-time variable, MV2_USE_APM (Section 11.63) can be enabled, as shown in the following example:

```
$ mpirun_rsh -np 2 n0 n1 MV2_USE_APM=1 ./cpi
```

or

```
$ mpiexec.hydra -n 2 -env MV2_USE_APM 1 ./cpi
```

MVAPICH2 also supports testing Automatic Path Migration in the subnet in the absence of network faults. This can be controlled by using a run-time variable MV2_USE_APM_TEST (Section 11.64). This should be combined with MV2_USE_APM as follows:

```
$ mpirun_rsh -np 2 n0 n1 MV2_USE_APM=1 MV2_USE_APM_TEST=1 ./cpi
```

or

```
$ mpiexec.hydra -n 2 -env MV2_USE_APM 1 -env MV2_USE_APM_TEST 1 ./cpi
```

6.8 Running with RDMA CM support (for OFA-IB-CH3 and OFA-iWARP-iWARP Devices)

In MVAPICH2, for using RDMA CM the runtime variable `MV2_USE_RDMA_CM` needs to be used as described in 11.

In addition to these flags, all the systems to be used need the following one time setup for enabling RDMA CM usage.

- **Setup the RDMA CM device:** RDMA CM device needs to be setup, configured with an IP address and connected to the network.

- **Setup the Local Address File:** Create the file (`/etc/mv2.conf`) with the local IP address to be used by RDMA CM. (Multiple IP addresses can be listed (one per line) for multirail configurations).
  ```
  $ echo 10.1.1.1 >> /etc/mv2.conf
  ```

Programs can be executed as follows:

```
$ mpirun_rsh -np 2 n0 n1 MV2_USE_RDMA_CM=1 prog
```

or

```
$ mpiexec.hydra -n 2 -env MV2_USE_RDMA_CM 1 prog
```

6.9 Running with Hot-Spot and Congestion Avoidance (for OFA-IB-CH3 Device)

MVAPICH2 supports hot-spot and congestion avoidance using InfiniBand multi-pathing mechanism. This support is available for MPI applications using OpenFabrics stack and InfiniBand adapters.

To enable this functionality, a run-time variable, MV2_USE_HSAM (Section 11.67) can be enabled, as shown in the following example:

```
$ mpirun_rsh -np 2 n0 n1 MV2_USE_HSAM=1 ./cpi
```

or

```
$ mpiexec.hydra -n 2 -env MV2_USE_HSAM 1 ./cpi
```

This functionality automatically defines the number of paths for hot-spot avoidance. Alternatively, the maximum number of paths to be used between a pair of processes can be defined by using a run-time variable MV2_NUM_QP_PER_PORT (Section 11.37).

We expect this functionality to show benefits in the presence of at least partially non-overlapping paths in the network. OpenSM, the subnet manager distributed with OpenFabrics supports LMC mechanism, which can be used to create multiple paths:

```
$ opensm -l4
```

will start the subnet manager with LMC value to four, creating sixteen paths between every pair of nodes.

7 Obtaining MVAPICH2 Library Version Information

The `mpiname` application is provided with MVAPICH2 to assist with determining the MPI library version and related information. The usage of `mpiname` is as follows:

```
mpiname [OPTION]
```

Print MPI library information. With no OPTION, the output is the same as -v.

-a print all information

-c print compilers

-d print device

-h display this help and exit

-n print the MPI name

-o print configuration options

-r print release date

-v print library version

8 Using OSU Benchmarks

If you have arrived at this point, you have successfully installed MVAPICH2. Congratulations!! In the `osu_benchmarks` directory, we provide these basic performance tests:

- One-way latency test (osu_latency.c)
- One-way bandwidth test (osu_bw.c)
- Bi-directional bandwidth (osu_bibw.c)
- Multiple Bandwidth / Message Rate test (osu_mbw_mr.c)
- One-sided put latency (osu_put_latency.c)
- One-sided put bandwidth (osu_put_bw.c)
- One-sided put bi-directional bandwidth (osu_put_bibw.c)
- One-sided get latency (osu_get_latency.c)
- One-sided get bandwidth (osu_get_bw.c)
- One-sided accumulate latency (osu_acc_latency.c)
- One-way multi-threaded latency test (osu_latency_mt.c) - Multi-threading support must be compiled in to run this test.
- Broadcast test (osu_bcast.c)
- Alltoall test (osu_alltoall.c)

The benchmarks are also periodically updated. The latest copy of the benchmarks can be downloaded from http://mvapich.cse.ohio-state.edu/benchmarks/. Sample performance numbers for these benchmarks on representative platforms with InfiniBand, iWARP and RoCE adapters are also included on our projects' web page. You are welcome to compare your performance numbers with our numbers. If you see any big discrepancy, please let us know by sending an email to mvapich-discuss@cse.ohio-state.edu.

9 Scalability features and Performance Tuning for Large Scale Clusters (Using CH3-based Interfaces)

MVAPICH2 provides many different parameters for tuning performance for a wide variety of platforms and applications. These parameters can be either compile time parameters or runtime parameters. Please refer to Section 9 for a complete description of all these parameters. In this section we classify these parameters depending on what you are tuning for and provide guidelines on how to use them.

9.1 Job Launch Tuning

Starting with version 1.2, MVAPICH2 has a new, scalable job launcher – mpirun_rsh which uses a tree based mechanism to spawn processes. The degree of this tree is determined dynamically to keep the depth low. For large clusters, it might be beneficial to further flatten the tree by specifying a higher degree. The degree can be overridden with the environment variable MV2_MT_DEGREE (see 11.33).

When running on large number of nodes, MVAPICH2 can use a faster, hierarchical launching mechanism. This mechanism can also be enabled manually by using the environment variable MV2_FASTSSH_THRESHOLD (see 11.18).

When the number of nodes involved is beyond 8k, the mpirun_rsh uses a file-based communication scheme to create the hierarchical tree. The default value can be overridden with the environment variable MV2_NPROCS_THRESHOLD (see 11.19).

9.2 Basic QP Resource Tuning

The following parameters affect memory requirements for each QP.

- MV2_DEFAULT_MAX_SEND_WQE
- MV2_DEFAULT_MAX_RECV_WQE
- MV2_MAX_INLINE_SIZE

MV2_DEFAULT_MAX_SEND_WQE and MV2_DEFAULT_MAX_RECV_WQE control the maximum number of WQEs per QP and MV2_MAX_INLINE_SIZE controls the maximum inline size. Reducing the values of these two parameters leads to less memory consumption. They are especially important for large scale clusters with a large amount of connections and multiple rails.

These two parameters are run-time adjustable. Please refer to Sections 11.13 and 11.29 for details.

9.3 RDMA Based Point-to-Point Tuning

The following parameters are important in tuning the memory requirements for adaptive rdma fast path feature.

- MV2_VBUF_TOTAL_SIZE
- MV2_NUM_RDMA_BUFFER
- MV2_RDMA_VBUF_POOL_SIZE

MV2_RDMA_VBUF_POOL_SIZE is a fixed number of pool of vbufs. These vbufs can be shared among all different connections depending on the communication needs of each connection.

On the other hand, the product of MV2_VBUF_TOTAL_SIZE and MV2_NUM_RDMA_BUFFER generally is a measure of the amount of memory registered for eager message passing. These buffers are not shared across connections.

In MVAPICH2, `MV2_VBUF_TOTAL_SIZE` is adjustable by environmental variables. Please refer to Section 11.86 for details.

9.4 Shared Receive Queue (SRQ) Tuning

The main environmental parameters controlling the behavior of the Shared Receive Queue design are:

- MV2_SRQ_SIZE (11.60)
- MV2_SRQ_LIMIT (11.59)

MV2_SRQ_SIZE is the maximum size of the Shared Receive Queue. You may increase this to value 1000 if the application requires very large number of processors (4K and beyond).
MV2_SRQ_LIMIT defines the low watermark for the flow control handler. This can be reduced if your aim is to reduce the number of interrupts.

9.5 eXtended Reliable Connection (XRC)

MVAPICH2 now supports the eXtended Reliable Connection (XRC) transport available in recent Mellanox HCAs. This transport helps reduce the number of QPs needed on multi-core systems. Set MV2_USE_XRC (11.83) to use XRC with MVAPICH2.

9.6 Shared Memory Tuning

MVAPICH2 uses shared memory communication channel to achieve high-performance message passing among processes that are on the same physical node. The two main parameters which are used for tuning shared memory performance for small messages are SMPI_LENGTH_QUEUE (11.88) and SMP_EAGER_SIZE (11.87). The two main parameters which are used for tuning shared memory performance for large messages are SMP_SEND_BUF_SIZE (11.90) and SMP_NUM_SEND_BUFFER (11.89).

SMPI_LENGTH_QUEUE is the size of the shared memory buffer which is used to store outstanding small and control messages. SMP_EAGER_SIZE defines the switch point from Eager protocol to Rendezvous protocol.

Messages larger than SMP_EAGER_SIZE are packetized and sent out in a pipelined manner. SMP_SEND_BUF_SIZE is the packet size, i.e. the send buffer size. SMP_NUM_SEND_BUFFER is the number of send buffers.

9.7 On-demand Connection Management Tuning

MVAPICH2 uses on-demand connection management to reduce the memory usage of MPI library. There are 4 parameters to tune connection manager: MV2_ON_DEMAND_THRESHOLD (11.39), MV2_CM_RECV_BUFFERS (11.7), MV2_CM_TIMEOUT (11.9), and MV2_CM_SPIN_COUNT (11.8). The first one applies to Gen2-IB, Gen2-iWARP and uDAPL devices and the other three only apply to Gen2 device.

MV2_ON_DEMAND_THRESHOLD defines threshold for enabling on-demand connection management scheme. When the size of the job is larger than the threshold value, on-demand connection management will be used.

MV2_CM_RECV_BUFFERS defines the number of buffers used by connection manager to establish new connections. These buffers are quite small and are shared for all connections, so this value may be increased to 8192 for large clusters to avoid retries in case of packet drops.

MV2_CM_TIMEOUT is the timeout value associated with connection management messages via UD channel. Decreasing this value may lead to faster retries but at the cost of generating duplicate messages.

MV2_CM_SPIN_COUNT is the number of the connection manager polls for new control messages from UD channel for each interrupt. This may be increased to reduce the interrupt overhead when many incoming control messages from UD channel at the same time.

9.8 Scalable Collectives Tuning

MVAPICH2 uses shared memory to get the best performance for many collective operations: MPI_Allreduce, MPI_Reduce, MPI_Barrier, and MPI_Bcast.

The important parameters for tuning these collectives are as follows. For MPI_Allreduce, the optimized shared memory algorithm is used until the MV2_SHMEM_ALLREDUCE_MSG (11.50).

Similarly for MPI_Reduce the corresponding threshold is MV2_SHMEM_REDUCE_MSG (11.56) and for MPI_BCAST the threshold can be set using MV2_SHMEM_BCAST_MSG (11.52). The default value for the SHMEM_BCAST_LEADERS parameter is set to 4K for this release.

The current version of MVAPICH2 also supports a 2-level point-to-point based Knomial algorithm for small messages in MPI_BCAST. The optimal threshold between this algorithm and the conventional binomial-tree algorithm varies across platforms and system sizes. Users can set the MV2_KNOMIAL_2LEVEL_BCAST_MESSAGE_SIZE_THRESHOLD (11.27) and the MV2_KNOMIAL_2LEVEL_BCAST_SYSTEM_SIZE_THRESHOLD (11.28) parameters to select the message size and the system size thresholds for using the knomial-based algorithm. With these settings, the normal binomial algorithm will be used for system sizes smaller than the chosen value for all small messages. For larger systems, the new knomial algorithm will be used for message sizes smaller than the value assigned to the MV2_KNOMIAL_2LEVEL_BCAST_MESSAGE_SIZE_THRESHOLD parameter. Users can also choose the inter-node and intra-node k-degree of the knomial bcast algorithm by using the parameters MV2_KNOMIAL_INTER_NODE_FACTOR (11.26) and MV2_KNOMIAL_INTRA_NODE_FACTOR (11.25). These values are currently set to 4.

10 FAQ and Troubleshooting with MVAPICH2

Based on our experience and feedback we have received from our users, here we include some of the problems a user may experience and the steps to resolve them. If you are experiencing any other problem, please feel free to contact us by sending an email to mvapich-discuss@cse.ohio-state.edu.

MVAPICH2 can be used over eight underlying interfaces, namely OFA-IB-CH3, OFA-IB-Nemesis, OFA-IWARP-CH3, OFA-RoCE-CH3, PSM-CH3, uDAPL-CH3, TCP/IP-CH3 and TCP/IP-Nemesis. Based on the underlying library being utilized, the troubleshooting steps may be different. However, some of the troubleshooting hints are common for all underlying libraries. Thus, in this section, we have divided the troubleshooting tips into four sections: General troubleshooting and Troubleshooting over any one of the five transport interfaces.

10.1 General Questions and Troubleshooting

10.1.1 Invalid Communicators Error

This is a problem which typically occurs due to the presence of multiple installations of MVAPICH2 on the same set of nodes. The problem is due to the presence of `mpi.h` other than the one, which is used for executing the program. This problem can be resolved by making sure that the `mpi.h` from other installation is not included.

10.1.2 Are `fork()` and `system()` supported?

`fork()` and `system()` is supported for the OpenFabrics device as long as the kernel is being used is Linux 2.6.16 or newer. Additionally, the version of OFED used should be 1.2 or higher. The environment variable IBV_FORK_SAFE=1 must also be set to enable fork support.

10.1.3 Cannot Build with the PathScale Compiler

There is a known bug with the PathScale compiler (before version 2.5) when building MVAPICH2. This problem will be solved in the next major release of the PathScale compiler. To work around this bug, use the the "`-LNO:simd=0`" C compiler option. This can be set in the build script similarly to:

```
export CC="pathcc -LNO:simd=0"
```

Please note the use of double quotes. If you are building shared libraries and are using the PathScale compiler (version below 2.5), then you should add "-g" to your CFLAGS, in order to get around a compiler bug.

10.1.4 MPI+OpenMP shows bad performance

MVAPICH2 uses CPU affinity to have better performance for single-threaded programs. For multi-threaded programs, e.g. MPI+OpenMP, it may schedule all the threads of a process to run on the same CPU. CPU affinity should be disabled in this case to solve the problem, i.e. set `-env MV2_ENABLE_AFFINITY 0`.

10.1.5 Error message "No such file or directory" when using Lustre file system

If you are using ADIO support for Lustre, please make sure that:
– Lustre is setup correctly, and that you are able to create, read to and write from files in the Lustre mounted directory.
– The Lustre directory is mounted on all nodes on which MVAPICH2 processes with ADIO support for Lustre are running.
– The path to the file is correctly specified.
– The permissions for the file or directory are correctly specified.

10.1.6 Program segfaults with "File locking failed in ADIOI_Set_lock"

If you are using ADIO support for Lustre, the recent Lustre releases require an additional mount option to have correct file locks.
So please include the following option with your lustre mount command: "-o localflock".
For example:

```
$ mount -o localflock -t lustre xxxx@o2ib:/datafs
/mnt/datafs
```

10.1.7 Running MPI programs built with gfortran

MPI programs built with gfortran might not appear to run correctly due to the default output buffering used by gfortran. If it seems there is an issue with program output, the `GFORTRAN_UNBUFFERED_ALL` variable can be set to "y" and exported into the environment before using the `mpiexec` command to launch the program, as done in the bash shell example below:

```
$ export GFORTRAN_UNBUFFERED_ALL=y
```

Or, if using `mpirun_rsh`, export the environment variable as in the example:

```
$ mpirun_rsh -np 2 n1 n2 GFORTRAN_UNBUFFERED_ALL=y ./a.out
```

10.1.8 Does MVAPICH2 work across AMD and Intel systems?

Yes, as long as you compile MVAPICH2 and your programs on one of the systems, either AMD or Intel, and run the same binary across the systems. MVAPICH2 has platform specific parameters for performance optimizations and it may not work if you compile MVAPICH2 and your programs on different systems and try to run the binaries together.

10.2 Failure with Job Launchers

10.2.1 /usr/bin/env: mpispawn: No such file or directory

If mpirun_rsh fails with this error message, it was unable to locate a necessary utility. This can be fixed by ensuring that all MVAPICH2 executables are in the PATH on all nodes.

If PATHs cannot be setup as mentioned, then invoke mpirun_rsh with a path prefix. For example:

```
/path/to/mpirun_rsh -np 2 node1 node2 ./mpi_proc
```

or

```
../../path/to/mpirun_rsh -np 2 node1 node2 ./mpi_proc
```

10.2.2 TotalView complains that "The MPI library contains no suitable type definition for struct MPIR_PROCDESC"

Ensure that the MVAPICH2 job launcher mpirun_rsh is compiled with debug symbols. Details are available in Section 5.2.12.

10.3 With OFA-IB-CH3 Interface

10.3.1 Cannot Open HCA

The above error reports that the InfiniBand Adapter is not ready for communication. Make sure that the drivers are up. This can be done by executing the following command which gives the path at which drivers are setup.

```
$ locate libibverbs
```

10.3.2 Checking state of IB Link

In order to check the status of the IB link, one of the following commands can be used:
`$ ibstatus`
or
`$ ibv_devinfo`.

10.3.3 Undefined reference to ibv_get_device_list

Add `-DGEN2_OLD_DEVICE_LIST_VERB` macro to CFLAGS and rebuild MVAPICH2-gen2. If this happens, this means that your Gen2 installation is old and needs to be updated.

10.3.4 Creation of CQ or QP failure

A possible reason could be inability to pin the memory required. Make sure the following steps are taken.

1. In `/etc/security/limits.conf` add the following

   ```
   * soft memlock phys_mem_in_KB
   ```

2. After this, add the following to `/etc/init.d/sshd`

   ```
   ulimit -l  phys_mem_in_KB
   ```

3. Restart sshd

With some distros, we've found that adding the ulimit -l line to the sshd init script is no longer necessary. For instance, the following steps work for our RHEL5 systems.

1. Add the following lines to `/etc/security/limits.conf`

   ```
   * soft memlock unlimited
   * hard memlock unlimited
   ```

2. Restart sshd

10.3.5 Hang with the HSAM Functionality

HSAM functionality uses multi-pathing mechanism with LMC functionality. However, some versions of OpenFabrics Drivers (including OpenFabrics Enterprise Distribution (OFED) 1.1) and using the Up*/Down* routing engine do not configure the routes correctly using the LMC mechanism. We strongly suggest to upgrade to OFED 1.2, which supports Up*/Down* routing engine and LMC mechanism correctly.

10.3.6 Failure with Automatic Path Migration

MVAPICH2 (OFA-IB-CH3) provides network fault tolerance with Automatic Path Migration (APM). However, APM is supported only with OFED 1.2 onwards. With OFED 1.1 and prior versions of OpenFabrics drivers, APM functionality is not completely supported. Please refer to Section 11.63 and section 11.64

10.3.7 Error opening file

If you configure MVAPICH2 with `RDMA_CM` and see this error, you need to verify if you have setup up the local IP address to be used by RDMA_CM in the file `/etc/mv2.conf`. Further, you need to make sure that this file has the appropriate file read permissions. Please follow Section 6.8 for more details on this.

10.3.8 RDMA CM Address error

If you get this error, please verify that the IP address specified /etc/mv2.conf is correctly specified with the IP address of the device you plan to use RDMA_CM with.

10.3.9 RDMA CM Route error

If see this error, you need to check whether the specified network is working or not.

10.3.10 TotalView does not seem to be working

If you are experiencing problems using TotalView, you should try this workaround:

```
$ ./configure ‘CFLAGS=-D_XOPEN_SOURCE=600’ .. your configure options ..
make && make install
```

10.4 With OFA-iWARP-CH3 Interface

10.4.1 Error opening file

If you configure MVAPICH2 with RDMA_CM and see this error, you need to verify if you have setup up the local IP address to be used by RDMA_CM in the file /etc/mv2.conf. Further, you need to make sure that this file has the appropriate file read permissions. Please follow Section 5.2.6 for more details on this.

10.4.2 RDMA CM Address error

If you get this error, please verify that the IP address specified /etc/mv2.conf is correctly specified with the IP address of the device you plan to use RDMA_CM with.

10.4.3 RDMA CM Route error

If see this error, you need to check whether the specified network is working or not.

10.5 With uDAPL-CH3 Interface

10.5.1 Cannot Open IA

If you configure MVAPICH2 with uDAPL and see this error, you need to check whether you have specified the correct uDAPL service provider (Section 5.2.8). If you have specified the uDAPL provider but still see this error, you need to check whether the specified network is working or not. If you are using OpenFabrics software stack on Linux, the default DAPL provider is `OpenIB-cma` for DAPL-1.2, and `ofa-v2-ib0` for DAPL-2.0. If you are using Solaris, the default DAPL provider is `ibd0`.

10.5.2 DAT Insufficient Resource

If you configure MVAPICH2 with uDAPL and see this error, you need to reduce the value of the environmental variables `RDMA_DEFAULT_MAX_SEND_WQE` and/or `RDMA_DEFAULT_MAX_RECV_WQE` depending on the underlying network.

10.5.3 Cannot Find libdat.so

If you get the error: "error while loading shared libraries, libdat.so", The location of the dat shared library is incorrect. You need to find the correct path of libdat.so and export `LD_LIBRARY_PATH` to this correct location. For example:

```
$ mpirun_rsh -np 2 n1 n2 LD_LIBRARY_PATH=/path/to/libdat.so ./a.out
```

or

```
$ export LD_LIBRARY_PATH=/path/to/libdat.so:$LD_LIBRARY_PATH
$ mpiexec -n 2 ./a.out
```

10.5.4 uDAPL over IB Does Not Scale Beyond 256 Nodes with rdma_cm Provider

We recommend that uDAPL IB consumers needing large scale-out use socket cm provider (libdaplscm.so) in leiu of rdma_cm (libdaplcma.so). iWARP users can remain using uDAPL rdma_cm provider. For detailed discussion of this issue please refer to:

`http://lists.openfabrics.org/pipermail/general/2008-June/051814.html`

10.6 Checkpoint/Restart

Please make sure the following things for a successful restart:

- The MPD must be started on all the compute nodes and the console node before a restart.
- The BLCR modules must be loaded on all the compute nodes and the console node before a restart
- The checkpoint file of MPI job console must be accessible from the console node.
- The corresponding checkpoint files of the MPI processes must be accessible from the compute nodes using the same path as when checkpoint was taken.

The following things can cause a restart to fail:

- The job which was checkpointed is not terminated or the some processes in that job are not cleaned properly. Usually they will be cleaned automatically, otherwise, since the pid can't be used by BLCR's to restart, restart will fail.
- The processes in the job have opened temporary files and these temporary files are removed or not accessible from the nodes where the processes are restarted on.

FAQ regarding Berkeley Lab Checkpoint/Restart (BLCR) can be found at `http://upc-bugs.lbl.gov/blcr/doc/html/FAQ.html` And the user guide for BLCR can be found at `http://upc-bugs.lbl.gov/blcr/doc/html/BLCR_Users_Guide.html`

If you encounter any problem with the Checkpoint/Restart support, please feel free to contact us as mvapich-discuss@cse.ohio-state.edu.

11 MVAPICH2 Parameters (CH3-Based Interfaces)

11.1 MV2_CKPT_FILE

- Class: Run Time
- Default: /tmp/ckpt
- Applicable interface(s): OFA-IB-CH3

This parameter specifies the path and the base filename for checkpoint files of MPI processes. The checkpoint files will be named as $MV2_CKPT_FILE.<number of checkpoint>.<process rank>, for example, /tmp/ckpt.1.0 is the checkpoint file for process 0's first checkpoint. To checkpoint on network-based file systems, user just need to specify the path to it, such as /mnt/pvfs2/my_ckpt_file.

11.2 MV2_CKPT_INTERVAL

- Class: Run Time
- Default: 0
- Unit: minutes
- Applicable interface(s): OFA-IB-CH3

This parameter can be used to enable automatic checkpointing. To let MPI job console automatically take checkpoints, this value needs to be set to the desired checkpointing interval. A zero will disable automatic checkpointing. Using automatic checkpointing, the checkpoint file for the MPI job console will be named as $MV2_CKPT_FILE.<number of checkpoint>.auto. Users need to use this file for restart.

11.3 MV2_CKPT_MAX_SAVE_CKPTS

- Class: Run Time
- Default: 0
- Applicable interface(s): OFA-IB-CH3

This parameter is used to limit the number of checkpoints saved on file system to save the file system space. When set to a positive value N, only the last N checkpoints will be saved.

11.4 MV2_CKPT_MPD_BASE_PORT

- Class: Run Time
- Default: 24678
- Applicable interface(s): OFA-IB-CH3

This parameter specifies the ports of socket connections to pass checkpointing control messages between MPD manager and MPI process. Users need to have a set of unused ports starting with $MV2_CKPT_MPD_BASE_PORT on the compute nodes. The used port will be the $MV2_CKPT_MPD_BASE_PORT + <process rank> for each MPI processes.

11.5 MV2_CKPT_MPIEXEC_PORT

- Class: Run Time
- Default: 14678
- Applicable interface(s): OFA-IB-CH3

This parameter specifies the port of the socket connection for passing checkpointing control messages on MPI job console node. Users need to have an unused port to be set to $MV2_CKPT_MPIEXEC_PORT on the console node.

11.6 MV2_CKPT_NO_SYNC

- Class: Run Time
- Applicable interface(s): OFA-IB-CH3

When this parameter is set to any value, the checkpoints will not be required to sync to disk. It can reduce the checkpointing delay in many cases. But if users are using local file system, or any parallel file system with local cache, to store the checkpoints, it is recommended not to set this parameter because otherwise the checkpoint files will be cached in local memory and will likely be lost upon failure.

11.7 MV2_CM_RECV_BUFFERS

- Class: Run Time
- Default: 1024
- Applicable interface(s): OFA-IB-CH3

This defines the number of buffers used by connection manager to establish new connections. These buffers are quite small and are shared for all connections, so this value may be increased to 8192 for large clusters to avoid retries in case of packet drops.

11.8 MV2_CM_SPIN_COUNT

- Class: Run Time
- Default: 5000
- Applicable interface(s): OFA-IB-CH3

This is the number of the connection manager polls for new control messages from UD channel for each interrupt. This may be increased to reduce the interrupt overhead when many incoming control messages from UD channel at the same time.

11.9 MV2_CM_TIMEOUT

- Class: Run Time
- Default: 500
- Unit: milliseconds
- Applicable interface(s): OFA-IB-CH3

This is the timeout value associated with connection management messages via UD channel. Decreasing this value may lead to faster retries but at the cost of generating duplicate messages.

11.10 MV2_CPU_MAPPING

- Class: Run Time
- Default: NA
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3 (Linux)

This allows users to specify process to CPU (core) mapping. The detailed usage of this parameter is described in Section 6.2.2. This parameter will not take effect if `MV2_ENABLE_AFFINITY` is set to 0, or if the library was configured with the "–disable-hwloc" option. MV2_CPU_MAPPING is currently not supported on Solaris.

11.11 MV2_CPU_BINDING_POLICY

- Class: Run Time
- Default: Bunch
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3 (Linux)

This allows users to specify process to CPU (core) mapping with the CPU binding policy. The detailed usage of this parameter is described in Section 6.2.1. This parameter will not take effect: if `MV2_ENABLE_AFFINITY` is set to 0; or `MV2_ENABLE_AFFINITY` is set to 1 and `MV2_CPU_MAPPING` is set, or if the library was configured with the "–disable-hwloc" option. The value of MV2_CPU_BINDING_POLICY can be "bunch" or "scatter". When this parameter takes effect and its value isn't set, "bunch" will be used as the default policy.

11.12 MV2_DAPL_PROVIDER

- Class: Run time
- Default: ofa-v2-ib0 (Linux DAPL v2.0), OpenIB-cma (Linux DAPL v1.2), ibd0 (Solaris)
- Applicable interface(s): uDAPL-CH3

This is to specify the underlying uDAPL-CH3 library that the user would like to use if MVAPICH2 is built with uDAPL-CH3.

11.13 MV2_DEFAULT_MAX_SEND_WQE

- Class: Run time
- Default: 64
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This specifies the maximum number of send WQEs on each QP. Please note that for OFA-IB-CH3 and OFA-iWARP-CH3, the default value of this parameter will be 16 if the number of processes is larger than 256 for better memory scalability.

11.14 MV2_DEFAULT_MAX_RECV_WQE

- Class: Run time
- Default: 128
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This specifies the maximum number of receive WQEs on each QP (maximum number of receives that can be posted on a single QP).

11.15 MV2_DEFAULT_MTU

- Class: Run time
- Default: OFA-IB-CH3: IBV_MTU_1024 for IB SDR cards and IBV_MTU_2048 for IB DDR and QDR cards. uDAPL-CH3: Network dependent.
- Applicable interface(s): OFA-IB-CH3, uDAPL-CH3

The internal MTU size. For OFA-IB-CH3, this parameter should be a string instead of an integer. Valid values are: `IBV_MTU_256`, `IBV_MTU_512`, `IBV_MTU_1024`, `IBV_MTU_2048`, `IBV_MTU_4096`.

11.16 MV2_DEFAULT_PKEY

- Class: Run Time
- Applicable device(s): OFA-IB-CH3

Select the partition to be used for the job.

11.17 MV2_ENABLE_AFFINITY

- Class: Run time
- Default: 1
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3 (Linux)

Enable CPU affinity by setting MV2_ENABLE_AFFINITY to 1 or disable it by setting MV2_ENABLE_AFFINITY to 0. MV2_ENABLE_AFFINITY is currently not supported on Solaris.

11.18 MV2_FASTSSH_THRESHOLD

- Class: Run time
- Default: 256
- Applicable device(s): All

Number of nodes beyond which to use hierarchical ssh during startup. This parameter is only relevant for mpirun_rsh based startup.

11.19 MV2_NPROCS_THRESHOLD

- Class: Run time
- Default: 8192
- Applicable device(s): All

Number of nodes beyond which to use file-based communication scheme in the hierarchical ssh during startup. This parameter is only relevant for mpirun_rsh based startup.

11.20 MV2_GET_FALLBACK_THRESHOLD

- Class: Run time
- This threshold value needs to be set in bytes.
- This option is effective if we define ONE_SIDED flag.
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This defines the threshold beyond which the MPI_Get implementation is based on direct one sided RDMA operations.

11.21 MV2_IBA_EAGER_THRESHOLD

- Class: Run time
- Default: Host Channel Adapter (HCA) dependent (12 KB for ConnectX HCA's)
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This specifies the switch point between eager and rendezvous protocol in MVAPICH2. For better performance, the value of MV2_IBA_EAGER_THRESHOLD should be set the same as MV2_VBUF_TOTAL_SIZE.

11.22 MV2_IBA_HCA

- Class: Run time
- Default: Unset
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This specifies the HCA to be used for performing network operations.

11.23 MV2_INITIAL_PREPOST_DEPTH

- Class: Run time
- Default: 10
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This defines the initial number of pre-posted receive buffers for each connection. If communication happen for a particular connection, the number of buffers will be increased to RDMA_PREPOST_DEPTH.

11.24 MV2_IWARP_MULTIPLE_CQ_THRESHOLD

- Class: Run time
- Default: 32
- Applicable interface(s): OFA-iWARP-CH3

This defines the process size beyond which we use multiple completion queues for iWARP interface.

11.25 MV2_KNOMIAL_INTRA_NODE_FACTOR

- Class: Run time
- Default: 4
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This defines the degree of the knomial operation during the intra-node knomial broadcast phase.

11.26 MV2_KNOMIAL_INTER_NODE_FACTOR

- Class: Run time
- Default: 4
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This defines the degree of the knomial operation during the inter-node knomial broadcast phase.

11.27 MV2_KNOMIAL_2LEVEL_BCAST_MESSAGE_SIZE_THRESHOLD

- Class: Run time
- Default: 2KB
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

When an intra-communicator MPI_Bcast operation is invoked with a communicator whose size is larger than the mv2_knomial_2level_bcast_system_size_threshold value, we use the 2-level knomial algorithm for message sizes smaller than the value set by this parameter.

11.28 MV2_KNOMIAL_2LEVEL_BCAST_SYSTEM_SIZE_THRESHOLD

- Class: Run time
- Default: 32
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

For communicator sizes larger than this value, the knomial bcast algorithm will be used for message sizes less than the value assigned to the mv2_knomial_2level_bcast_message_size_threshold parameter. For smaller systems, we use the default binomial tree based algorithm.

11.29 MV2_MAX_INLINE_SIZE

- Class: Run time
- Default: Network card dependent (128 for most networks including InfiniBand)
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This defines the maximum inline size for data transfer. Please note that the default value of this parameter will be 0 when the number of processes is larger than 256 to improve memory usage scalability.

11.30 MV2_MAX_NUM_WIN

- Class: Run time
- Default: 16
- Applicable interface(s): OFA-IB-CH3

Maximum number of RMA windows that can be created and active concurrently. Typically this value is sufficient for most applications. Increase this value to the number of windows your application uses

11.31 MV2_MPD_RECVTIMEOUT_MULTIPLIER

- Class: Run time
- Default: 0.05

The multiplier to be added to the MPD mpiexec timeout for each process in a job.

11.32 MV2_MPIRUN_TIMEOUT

- Class: Run time
- Default: Dynamic - based on number of nodes

The number of seconds after which mpirun_rsh aborts job launch. Note that unlike most other parameters described in this section, this is an environment variable that has to be set in the runtime environment (for e.g. through export in the bash shell).

11.33 MV2_MT_DEGREE

- Class: Run time
- Default: Dynamic - based on number of nodes

The degree of the hierarchical tree used by mpirun_rsh. By default mpirun_rsh uses a value that tries to keep the depth of the tree to 4. Note that unlike most other parameters described in this section, this is an environment variable that has to be set in the runtime environment (for e.g. through export in the bash shell).

11.34 MV2_NDREG_ENTRIES

- Class: Run time
- Default: 1000
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This defines the total number of buffers that can be stored in the registration cache. It has no effect if MV2_USE_LAZY_MEM_UNREGISTER is not set. A larger value will lead to less frequent lazy de-registration.

11.35 MV2_NUM_HCAS

- Class: Run time
- Default: 1
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter indicates number of InfiniBand adapters to be used for communication on an end node.

11.36 MV2_NUM_PORTS

- Class: Run time
- Default: 1
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter indicates number of ports per InfiniBand adapter to be used for communication per adapter on an end node.

11.37 MV2_NUM_QP_PER_PORT

- Class: Run time
- Default: 1
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter indicates number of queue pairs per port to be used for communication on an end node. This is useful in the presence of multiple send/recv engines available per port for data transfer.

11.38 MV2_NUM_RDMA_BUFFER

- Class: Run time
- Default: Architecture dependent (32 for EM64T)
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

The number of RDMA buffers used for the RDMA fast path. This *fast path* is used to reduce latency and overhead of small data and control messages. This value will be ineffective if MV2_USE_RDMA_FAST_PATH is not set.

11.39 MV2_ON_DEMAND_THRESHOLD

- Class: Run Time
- Default: 64 (OFA-IB-CH3, uDAPL-CH3), 16 (OFA-iWARP-CH3)
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This defines threshold for enabling on-demand connection management scheme. When the size of the job is larger than the threshold value, on-demand connection management will be used.

11.40 MV2_PREPOST_DEPTH

- Class: Run time
- Default: 64
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This defines the number of buffers pre-posted for each connection to handle send/receive operations.

11.41 MV2_PSM_DEBUG

- Class: Run time (Debug)
- Default: 0
- Applicable interface: PSM

This parameter enables the dumping of run-time debug counters from the MVAPICH2-PSM progress engine. Counters are dumped every PSM_DUMP_FREQUENCY seconds.

11.42 MV2_PSM_DUMP_FREQUENCY

- Class: Run time (Debug)
- Default: 10 seconds
- Applicable interface: PSM

This parameters sets the frequency for dumping MVAPICH2-PSM debug counters. Value takes effect only in PSM_DEBUG is enabled.

11.43 MV2_PUT_FALLBACK_THRESHOLD

- Class: Run time
- This threshold value needs to be set in bytes.
- This option is effective if we define ONE_SIDED flag.
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This defines the threshold beyond which the MPI_Put implementation is based on direct one sided RDMA operations.

11.44 MV2_RDMA_CM_ARP_TIMEOUT

- Class: Run Time
- Default: 2000 ms
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter specifies the arp timeout to be used by RDMA CM module.

11.45 MV2_RDMA_CM_MAX_PORT

- Class: Run Time
- Default: Unset
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter specifies the upper limit of the port range to be used by the RDMA CM module when choosing the port on which it listens for connections.

11.46 MV2_RDMA_CM_MIN_PORT

- Class: Run Time
- Default: Unset
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter specifies the lower limit of the port range to be used by the RDMA CM module when choosing the port on which it listens for connections.

11.47 MV2_RNDV_PROTOCOL

- Class: Run time
- Default: RPUT
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

The value of this variable can be set to choose different Rendezvous protocols. RPUT (default RDMA-Write) RGET (RDMA Read based), R3 (send/recv based).

11.48 MV2_R3_THRESHOLD

- Class: Run time
- Default: 4096
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

The value of this variable controls what message sizes go over the R3 rendezvous protocol. Messages above this message size use MV2_RNDV_PROTOCOL.

11.49 MV2_R3_NOCACHE_THRESHOLD

- Class: Run time
- Default: 32768
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

The value of this variable controls what message sizes go over the R3 rendezvous protocol when the registration cache is disabled (MV2_USE_LAZY_MEM_UNREGISTER=0). Messages above this message size use MV2_RNDV_PROTOCOL.

11.50 MV2_SHMEM_ALLREDUCE_MSG

- Class: Run Time
- Default: $1 \ll 15$
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

The SHMEM allreduce is used for messages less than this threshold.

11.51 MV2_SHMEM_BCAST_LEADERS

- Class: Run time
- Default: 4096

The number of leader processes that will take part in the SHMEM broadcast operation. Must be greater than the number of nodes in the job.

11.52 MV2_SHMEM_BCAST_MSG

- Class: Run Time
- Default: $1 \ll 20$
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

The SHMEM bcast is used for messages less than this threshold.

11.53 MV2_SHMEM_COLL_MAX_MSG_SIZE

- Class: Run Time
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter can be used to select the max buffer size of message for shared memory collectives.

11.54 MV2_SHMEM_COLL_NUM_COMM

- Class: Run Time
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter can be used to select the number of communicators using shared memory collectives.

11.55 MV2_SHMEM_DIR

- Class: Run Time
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3
- Default: /dev/shm for Linux and /tmp for Solaris

This parameter can be used to specify the path to the shared memory files for intra-node communication.

11.56 MV2_SHMEM_REDUCE_MSG

- Class: Run Time
- Default: $1 \ll 10$
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

The SHMEM reduce is used for messages less than this threshold.

11.57 MV2_SM_SCHEDULING

- Class: Run Time
- Default: USE_FIRST (Options: ROUND_ROBIN)
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

11.58 MV2_SMP_USE_LIMIC2

- Class: Run Time
- Default: On if configured with –with-limic2
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter enables/disables LiMIC2 at run time. It does not take effect if MVAPICH2 is not configured with –with-limic2.

11.59 MV2_SRQ_LIMIT

- Class: Run Time
- Default: 30
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This is the low watermark limit for the Shared Receive Queue. If the number of available work entries on the SRQ drops below this limit, the flow control will be activated.

11.60 MV2_SRQ_SIZE

- Class: Run Time
- Default: 512

- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This is the maximum number of work requests allowed on the Shared Receive Queue.

11.61 MV2_STRIPING_THRESHOLD

- Class: Run Time
- Default: 8192
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter specifies the message size above which we begin the stripe the message across multiple rails (if present).

11.62 MV2_SUPPORT_DPM

- Class: Run time
- Default: 0 (disabled)
- Applicable interface: OFA-IB-CH3

This option enables the dynamic process management interface and on-demand connection management.

11.63 MV2_USE_APM

- Class: Run Time
- Applicable interface(s): OFA-IB-CH3

This parameter is used for recovery from network faults using Automatic Path Migration. This functionality is beneficial in the presence of multiple paths in the network, which can be enabled by using lmc mechanism.

11.64 MV2_USE_APM_TEST

- Class: Run Time
- Applicable interface(s): OFA-IB-CH3

This parameter is used for testing the Automatic Path Migration functionality. It periodically moves the alternate path as the primary path of communication and re-loads another alternate path.

11.65 MV2_USE_BLOCKING

- Class: Run time
- Default: 0
- Applicable interface(s): OFA-IB-CH3

Setting this parameter enables mvapich2 to use blocking mode progress. MPI applications do not take up any CPU when they are waiting for incoming messages.

11.66 MV2_USE_COALESCE

- Class: Run time
- Default: set
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

Setting this parameter enables message coalescing to increase small message throughput

11.67 MV2_USE_HSAM

- Class: Run Time
- Applicable interface(s): OFA-IB-CH3

This parameter is used for utilizing hot-spot avoidance with InfiniBand clusters. To leverage this functionality, the subnet should be configured with lmc greater than zero. Please refer to section 6.9 for detailed information.

11.68 MV2_USE_IWARP_MODE

- Class: Run Time
- Default: unset
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter enables the library to run in iWARP mode. The library has to be built using the flag -DRDMA_CM for using this feature.

11.69 MV2_USE_KNOMIAL_2LEVEL_BCAST

- Class: Run time
- Default: 1
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3 (Linux)

Enable Knomial Broadcast by setting MV2_USE_KNOMIAL_2LEVEL_BCAST to 1 or disable it by setting
MV2_USE_KNOMIAL_2LEVEL_BCAST to 0. The other knomial related variables are :

- MV2_KNOMIAL_INTRA_NODE_FACTOR
- MV2_KNOMIAL_INTER_NODE_FACTOR
- MV2_KNOMIAL_2LEVEL_BCAST_THRESHOLD

11.70 MV2_USE_LAZY_MEM_UNREGISTER

- Class: Run time
- Default: set
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

Setting this parameter enables mvapich2 to use memory registration cache.

11.71 MV2_USE_RDMAOE

- Class: Run Time
- Default: Un Set
- Applicable interface(s): OFA-IB-CH3

This parameter enables the use of RDMA over Ethernet for MPI communication. The underlying HCA and network must support this feature.

11.72 MV2_USE_RDMA_CM

- Class: Run Time
- Default: Network Dependant (set for OFA-iWARP-CH3 and unset for OFA-IB-CH3)
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

This parameter enables the use of RDMA CM for establishing the connections. The library has to be built using the flag -DRDMA_CM for using this feature.

11.73 MV2_USE_RDMA_FAST_PATH

- Class: Run time
- Default: set
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

Setting this parameter enables mvapich2 to use adaptive rdma fast path features for OFA-IB-CH3 interface and static rdma fast path features for uDAPL-CH3 interface.

11.74 MV2_USE_RDMA_ONE_SIDED

- Class: Run time
- Default: set
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

Setting this parameter allows mvapich2 to use optimized one sided implementation based RDMA operations.

11.75 MV2_USE_RING_STARTUP

- Class: Run time
- Default: set
- Applicable interface(s): OFA-IB-CH3

Setting this parameter enables mvapich2 to use ring based startup.

11.76 MV2_USE_SHARED_MEM

- Class: Run time
- Default: set
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

Use shared memory for intra-node communication.

11.77 MV2_USE_SHMEM_ALLREDUCE

- Class: Run Time
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3, VAPI

This parameter can be used to turn off shared memory based MPI_Allreduce for OFA-IB-CH3 over IBA by setting this to 0.

11.78 MV2_USE_SHMEM_BARRIER

- Class: Run Time
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3, VAPI

This parameter can be used to turn off shared memory based MPI_Barrier for OFA-IB-CH3 over IBA by setting this to 0.

11.79 MV2_USE_SHMEM_BCAST

- Class: Run Time
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This parameter can be used to turn off shared memory based MPI_Bcast for OFA-IB-CH3 over IBA by setting this to 0.

11.80 MV2_USE_SHMEM_COLL

- Class: Run time
- Default: set
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

Use shared memory for collective communication. Set this to 0 for disabling shared memory collectives.

11.81 MV2_USE_SHMEM_REDUCE

- Class: Run Time
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3, VAPI

This parameter can be used to turn off shared memory based MPI_Reduce for OFA-IB-CH3 over IBA by setting this to 0.

11.82 MV2_USE_SRQ

- Class: Run time
- Default: set
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

Setting this parameter enables mvapich2 to use shared receive queue.

11.83 MV2_USE_XRC

- Class: Run time
- Default: 0
- Applicable device(s): OFA-IB-CH3

Use the XRC InfiniBand transport available since Mellanox ConnectX adapters. This features requires OFED version later than 1.3. It also automatically enables SRQ and ON-DEMAND connection management. Note that the MVAPICH2 library needs to have been configured with –enable-xrc=yes to use this feature.

11.84 MV2_VBUF_POOL_SIZE

- Class: Run time
- Default: 2048
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

The number of vbufs in the initial pool. This pool is shared among all the connections.

11.85 MV2_VBUF_SECONDARY_POOL_SIZE

- Class: Run time
- Default: 256
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

The number of vbufs allocated each time when the global pool is running out in the initial pool. This is also shared among all the connections.

11.86 MV2_VBUF_TOTAL_SIZE

- Class: Run time
- Default: Host Channel Adapter (HCA) dependent (12 KB for ConnectX HCA's)
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3

The size of each `vbuf`, the basic communication buffer of MVAPICH2. For better performance, the value of MV2_IBA_EAGER_THRESHOLD should be set the same as MV2_VBUF_TOTAL_SIZE.

11.87 SMP_EAGERSIZE

- Class: Run time
- Default: Architecture dependent
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This parameter defines the switch point from Eager protocol to Rendezvous protocol for intra-node communication. Note that this variable should be set in KBytes.

11.88 SMPI_LENGTH_QUEUE

- Class: Run time
- Default: Architecture dependent
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This parameter defines the size of shared buffer between every two processes on the same node for transferring messages smaller than or equal to SMP_EAGERSIZE. Note that this variable should be set in KBytes.

11.89 SMP_NUM_SEND_BUFFER

- Class: Run time
- Default: Architecture dependent
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This parameter defines the number of internal send buffers for sending intra-node messages larger than SMP_EAGERSIZE.

11.90 SMP_SEND_BUF_SIZE

- Class: Compile time
- Default: Architecture dependent
- Applicable interface(s): OFA-IB-CH3, OFA-iWARP-CH3, uDAPL-CH3

This parameter defines the packet size when sending intra-node messages larger than SMP_EAGERSIZE.

12 MVAPICH2 Parameters (OFA-IB-Nemesis Interface)

12.1 MV2_DEFAULT_MAX_SEND_WQE

- Class: Run time
- Default: 64

This specifies the maximum number of send WQEs on each QP. Please note that for Gen2 and Gen2-iWARP, the default value of this parameter will be 16 if the number of processes is larger than 256 for better memory scalability.

12.2 MV2_DEFAULT_MAX_RECV_WQE

- Class: Run time
- Default: 128

This specifies the maximum number of receive WQEs on each QP (maximum number of receives that can be posted on a single QP).

12.3 MV2_DEFAULT_MTU

- Class: Run time
- Default: IBV_MTU_1024 for IB SDR cards and IBV_MTU_2048 for IB DDR and QDR cards. uDAPL: Network dependent.

The internal MTU size. For Gen2, this parameter should be a string instead of an integer. Valid values are: `IBV_MTU_256`, `IBV_MTU_512`, `IBV_MTU_1024`, `IBV_MTU_2048`, `IBV_MTU_4096`.

12.4 MV2_DEFAULT_PKEY

- Class: Run Time

Select the partition to be used for the job.

12.5 MV2_IBA_EAGER_THRESHOLD

- Class: Run time
- Default: Architecture dependent (12KB for IA-32)

This specifies the switch point between eager and rendezvous protocol in MVAPICH2. For better performance, the value of MV2_IBA_EAGER_THRESHOLD should be set the same as MV2_VBUF_TOTAL_SIZE.

12.6 MV2_INITIAL_PREPOST_DEPTH

- Class: Run time
- Default: 10

This defines the initial number of pre-posted receive buffers for each connection. If communication happen for a particular connection, the number of buffers will be increased to
RDMA_PREPOST_DEPTH.

12.7 MV2_MAX_INLINE_SIZE

- Class: Run time
- Default: Network card dependent (128 for most networks including InfiniBand)

This defines the maximum inline size for data transfer. Please note that the default value of this parameter will be 0 when the number of processes is larger than 256 to improve memory usage scalability.

12.8 MV2_MPIRUN_TIMEOUT

- Class: Run time
- Default: Dynamic - based on number of nodes

The number of seconds after which mpirun_rsh aborts job launch. Note that unlike most other parameters described in this section, this is an environment variable that has to be set in the runtime environment (for e.g. through export in the bash shell).

12.9 MV2_MT_DEGREE

- Class: Run time
- Default: Dynamic - based on number of nodes

The degree of the hierarchical tree used by mpirun_rsh. By default mpirun_rsh uses a value that tries to keep the depth of the tree to 4. Note that unlike most other parameters described in this section, this is an environment variable that has to be set in the runtime environment (for e.g. through export in the bash shell).

12.10 MV2_NDREG_ENTRIES

- Class: Run time
- Default: 1000

This defines the total number of buffers that can be stored in the registration cache. It has no effect if MV2_USE_LAZY_MEM_UNREGISTER is not set. A larger value will lead to less frequent lazy de-registration.

12.11 MV2_NUM_RDMA_BUFFER

- Class: Run time
- Default: Architecture dependent (32 for EM64T)

The number of RDMA buffers used for the RDMA fast path. This *fast path* is used to reduce latency and overhead of small data and control messages. This value will be ineffective if MV2_USE_RDMA_FAST_PATH is not set.

12.12 MV2_PREPOST_DEPTH

- Class: Run time
- Default: 64

This defines the number of buffers pre-posted for each connection to handle send/receive operations.

12.13 MV2_RNDV_PROTOCOL

- Class: Run time
- Default: RPUT

The value of this variable can be set to choose different Rendezvous protocols. RPUT (default RDMA-Write) RGET (RDMA Read based), R3 (send/recv based).

12.14 MV2_R3_THRESHOLD

- Class: Run time
- Default: 4096

The value of this variable controls what message sizes go over the R3 rendezvous protocol. Messages above this message size use MV2_RNDV_PROTOCOL.

12.15 MV2_R3_NOCACHE_THRESHOLD

- Class: Run time
- Default: 32768

The value of this variable controls what message sizes go over the R3 rendezvous protocol when the registration cache is disabled (MV2_USE_LAZY_MEM_UNREGISTER=0). Messages above this message size use MV2_RNDV_PROTOCOL.

12.16 MV2_SRQ_LIMIT

- Class: Run Time
- Default: 30

This is the low watermark limit for the Shared Receive Queue. If the number of available work entries on the SRQ drops below this limit, the flow control will be activated.

12.17 MV2_SRQ_SIZE

- Class: Run Time
- Default: 512

This is the maximum number of work requests allowed on the Shared Receive Queue.

12.18 MV2_STRIPING_THRESHOLD

- Class: Run Time
- Default: 8192

This parameter specifies the message size above which we begin the stripe the message across multiple rails (if present).

12.19 MV2_USE_BLOCKING

- Class: Run time
- Default: 0

Setting this parameter enables mvapich2 to use blocking mode progress. MPI applications do not take up any CPU when they are waiting for incoming messages.

12.20 MV2_USE_LAZY_MEM_UNREGISTER

- Class: Run time
- Default: set

Setting this parameter enables mvapich2 to use memory registration cache.

12.21 MV2_USE_RDMA_FAST_PATH

- Class: Run time
- Default: set

Setting this parameter enables mvapich2 to use adaptive rdma fast path features for Gen2 interface and static rdma fast path features for uDAPL interface.

12.22 MV2_USE_SRQ

- Class: Run time
- Default: set

Setting this parameter enables mvapich2 to use shared receive queue.

12.23 MV2_VBUF_POOL_SIZE

- Class: Run time
- Default: 512

The number of vbufs in the initial pool. This pool is shared among all the connections.

12.24 MV2_VBUF_SECONDARY_POOL_SIZE

- Class: Run time
- Default: 128

The number of vbufs allocated each time when the global pool is running out in the initial pool. This is also shared among all the connections.

12.25 MV2_VBUF_TOTAL_SIZE

- Class: Run time
- Default: Architecture dependent (6 KB for EM64T)

The size of each `vbuf`, the basic communication buffer of MVAPICH2. For better performance, the value of MV2_IBA_EAGER_THRESHOLD should be set the same as MV2_VBUF_TOTAL_SIZE.